Evaluation und Optimierung der automatischen Sachgruppenvergabe mithilfe des HDLTex Tools

Untersuchung des Ansatzes, ob eine Kombination der Vorhersagen aus den Hierarchieebenen, zu einer Verbesserung der Textklassifizierung führt

Master-Thesis

zur Erlangung des Grades

Master of Science

Im Studiengang Business Application Architecture Master an der Fakultät Wirtschaftsinformatik der Hochschule Furtwangen University

Referent:	Prof. Dr. Andreas Heß
Korreferent:	Prof. Dr. Holger Ziekow
Vorgelegt am:	31.08.2018
Vorgelegt von:	Alexander Gerling
	Hauensteiner Str. 2
	79774 Albbruck
	Matrikelnummer: 256185
	alexander.gerling@hs-furtwangen.de

Abstract

In this thesis, new methods for text classification are examined and compared to the current software of the DNB. Due to technical changes in the area machine learning in recent years, improvements in text classification have been achieved. The objective is to improve the subject groups allocation of the DNB and to allow a hierarchical classification based on the DDC system. The decision was made on the HDLTex tool, as the structure of the DNB data set and the DDC system, which supports a hierarchical classification, are perfectly designed for it. The use of RNN networks on both hierarchical levels improved the current software situation. Furthermore, the approach was examined, if a combination of the predictions of the two hierarchies levels leads to an additional improvement, which, however, produced a negative result. Both beginners and experts find themselves as readers of this master's thesis in the target group again.

In dieser Arbeit werden neue Verfahren zur Textklassifizierung untersucht und der aktuellen Software der DNB gegenübergestellt. Durch technische Veränderungen im Bereich Machine Learning in den letzten Jahren, konnten Verbesserungen in der Textklassifizierung erzielt werden. Dabei soll die Sachgruppenvergabe der DNB verbessert und anhand des DDC Systems eine hierarchische Klassifizierung ermöglicht werden. Die Entscheidung fiel auf das HDLTex Tool, da die Struktur des Datensatzes der DNB und das DDC System, welche eine hierarchische Klassifizierung unterstützt, perfekt darauf ausgelegt sind. Durch die Nutzung von RNN Netzen auf beiden Hierarchieebenen konnte eine Verbesserung zu der aktuellen Software erzielt werden. Weiterhin wurde der Ansatz untersucht, ob eine Kombinierung der Vorhersagen der beiden Hierarchieebenen zu einer aufbauenden Verbesserung führt, welches jedoch ein negatives Ergebnis hervorbrachte. Sowohl Anfänger als auch Experten finden sich als Leser dieser Masterarbeit in der Zielgruppe wieder.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

SVM	Support Vector Machine
DNB	Deutsche Nationalbibliothek
PoS	Part of Speech
DDC	Dewey-Dezimalklassifikation
HDLTex	Hierarchical Deep Learning for Text classification
RMDL	Random Multimodel Deep Learning for Classification
NLP	Natural Language Processing
BoW	Bag of Words
DBN	Deep Belief Networks
RBM	Restricted Boltzmann Machines
MLP	Multilayer Perceptron
DNN	Deep Neural Network
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
TF	Term Frequency
TF-IDF	Term Frequency - Inverse Document Frequency
GloVe	Global Vectors for Word Representation

1 Einleitung

Die Digitalisierung schreitet immer weiter voran. Nach einer Umfrage der International Data Corporation im Jahr 2016 waren 47% aller Dokumente in deutschen Büros elektronischer Natur [1]. Nicht nur Arbeitsplätze in großen Unternehmen oder Universitäten steigen auf den Trend papierloses Arbeiten um, sondern auch die Deutsche Nationalbibliothek (DNB). Am 22. Juni 2006 wurde das Gesetz für die DNB verabschiedet, dass die DNB zur Sammlung, Erschließung, Verzeichnung und Langzeitarchivierung von unkörperlichen Medienwerken wie beispielsweise einer Netzpublikation verpflichtet [2]. Momentan sind an der DNB über vier Millionen Netzpublikationen archiviert und es stehen noch weitere Millionen digitale Medien aus. Durch das anhaltende Wachstum wird für das Jahr 2021 eine Anzahl von 26 Millionen zu archivierenden Publikationen vorhergesagt [2]. Somit steht die DNB vor der riesigen Aufgabe, die stetig wachsende Anzahl von Netzpublikationen inhaltlich richtig zu erschließen und zu kategorisieren. Im Jahr 2016 wurden 613 Netzpublikationen gezählt und im folgenden Jahr waren es bereits 1372 Netzpublikationen. Somit sind im Jahr 2017 doppelt so viele Netzpublikationen als im Vorjahr eingegangen [3]. Täglich gehen an die 1400 Objekte in der DNB ein. Diese Zahl stammt aus dem Jahre 2015 und es ist mit einem Anstieg über die vergangenen Jahre zu rechnen. Weiterhin sammelt die DNB Webseiten, Monografien, E-Paper, E-Books, E-Journals, Noten und Hörbücher [4]. Die intellektuelle inhaltliche Erschließung der Netzpublikationen ist somit nur noch mithilfe maschineller Unterstützung zu bewältigen. Dafür werden neue Verfahren, Workflows und Prozesse entwickelt, um dem Anspruch gerecht zu werden. Auch die Erweiterung der IT-Infrastruktur muss vorangetrieben werden, um die Maße an Daten verarbeiten zu können [2].

Beim Hochladen einer Netzpublikation werden dem Dokument automatisch Schlagwörter vergeben und mehrere Sachgruppen hinzugefügt. Mit dem Projekt PETRUS verfolgte die DNB das Ziel, deutschsprachige Netzpublikationen automatisch mit Schlagwörtern anzureichern. Das wird mithilfe eines kontrollierten Vokabulars auf Grundlage von Volltexten, Abstracts oder Inhaltsverzeichnissen ermöglicht. Für diese Aufgabe hat sich die DNB über eine Ausschreibung für die Software *Averbis Extraction Platform* des Unternehmens

Averbis entschieden [5]. Als Teilprojekt innerhalb des Projektes PETRUS, wurde die automatische Sachgruppenvergabe anhand der Dewey-Dezimalklassifikation (DDC) untersucht. Ziel dabei ist es, Netzpublikationen bis zu drei Sachgruppen zu vergeben, falls eine eindeutige Zuweisung einer Sachgruppe nicht möglich ist. Die maschinelle Vergabe der DDC Sachgruppen, wurde mit Abschluss des Projektes PETRUS in den Produktionsbetrieb übernommen [6].

1.1 Problembeschreibung

Die momentane Sachgruppenvergabe anhand des DDC Systems, funktioniert mit der verwendeten Software der DNB recht solide und zuverlässig. Durch technische Neuerungen in den letzten Jahren im Bereich Machine Learning, soll überprüft werden, ob mögliche Verbesserungen implementiert werden können. Um die Problemstellung bei der Sachgruppenvergabe zu demonstrieren, soll ein Beispiel aus der Vergangenheit helfen. Dabei wird der Titel: *Java Programmierung im Bereich Internet of Things – Verwendung eines Raspberry Pi zur Erstellung von digitalen Einkaufslisten – Kann mit Hilfe eines Raspberry Pi das Einkaufen revolutioniert werden?* [7] von Eduard Franke genommen. Dieser Titel wurde in die Sachgruppen *004 Informatik* und *640 Hauswirtschaft und Familienleben* aufgenommen. Die erste Sachgruppe stimmt mit dem Inhalt der Arbeit überein, jedoch verfehlt die zweite Angabe den Themenbereich. Hier kann ganz konkret der Wunsch nach Verbesserung aufgeführt werden. Es stellt sich jedoch die Frage, wie es zu dieser falschen Klassifizierung kommen konnte. An dieser Stelle könnte spekuliert werden, ob das Wort *Einkaufslisten* im Titel zu der falschen Klassifizierung führte. Problematisch sind die Klassifizierungen von kurzen Texten, da diese nur eine begrenzte Anzahl von ausschlaggebenden Wörtern besitzt. Anhand des oben genannten Titels könnten folgende Wörter zur Klassifizierung beitragen: *Java Programmierung*, *Internet of Things*, *Raspberry Pi*, *digitalen*, *Einkaufslisten*, *Einkaufen*, *revolutioniert*. Bei dieser Publikation standen sowohl der Volltext, das Abstract und das Inhaltsverzeichnis zur Verfügung. Trotz der weiteren Quellen, die zur Klassierung genutzt werden konnten, wurde nur der Titel verwendet. Dabei entstand das Problem, dass zu wenige ausschlaggebende Wörter vorhanden waren

und dabei die richtige Klassifizierung beeinträchtigten. Weiterhin ist entscheidend, wie die Wörter verwendet werden. Als Beispiel könnte *Internet of Things* vielfach interpretiert werden. Es könnten etwa die einzelnen Wörter, wie *Internet*, *of*, *Things* oder vermischte Wörter *Internet of*, *of Things* genutzt werden. Bei dieser Problematik sollte das Vorverarbeiten der Daten oder im Einzelfall einer Instanz, nicht außer Acht gelassen werden. Unterschiedliche Methoden zur Vorbereitung der Daten werden im Kapitel 3 genauer betrachtet. Weiterhin wird beschrieben, wie die einzelnen Features zur Klassifizierung beitragen.

Der obere Teil dieses Unterkapitels, soll dazu dienen die Oberflächliche beziehungsweise, die sofort ersichtliche Problematik zu beschreiben. Jedoch sind, dass nicht die einzigen relevanten Aspekte, die zu beachten sind. Beim Trainieren des zu vorhersagenden Modells müssen wichtige Elemente betrachtet werden. Hierfür konnten die originalen Titeldaten der DNB herangezogen werden, welche in einem Datensatz zusammengefügt wurden. Dieser Datensatz beinhalten 80 Sachgruppen, welche als Labels für eine hierarchische Verteilung auf zwei Ebenen dienen soll. Als Beispiel soll das Label 07 dienen. Dabei muss erwähnt werden, dass das Label 07 nicht mehr der originalen DDC Sachgruppierung entspricht und aus den Labels 070 – 079 entstand. Diese Labels wurden aus Zweckmäßigkeit auf zwei Stellen gekürzt. Die Zahl 0 stellt die erste Hierarchieebene dar und gehört zu der Hauptkategorie *Informatik, Informationswissenschaft & allgemeine Werke*. Die zweite Zahl 7, gehört zu der untergeordneten Kategorie *Nachrichtenmedien, Journalismus, Verlagswesen*. Dazu kommt, dass die 80 Labels von den Instanzen überaus ungleichmäßig auf die Sachgruppen verteilt sind. Bei Hochschulschriften werden an die 90 % der Publikationen auf ungefähr 20 % der vorhanden Sachgruppen verteilt [8]. Diese Tatsache verschlechtert die richtige Vorhersage der Klassen und stellt ein großes Problem dar. Da es nicht möglich ist, weitere Instanzen zu beschaffen, muss eine andere Lösung gefunden werden. Hierfür muss für die Instanzen innerhalb des Datensatzes eine gewisse Qualität festgelegt werden.

1.2 Vorgehensweise

In Unterkapitel 1.1 wurde die Problemstellung aufgeführt und anhand dieser Problematiken müssen Technologien gefunden werden, die den Stand der Technik wiedergeben und optimalerweise eine Verbesserung herbeiführen.

Dazu wird die Literaturrecherche auf anwendbare Open Source Software, als auch auf State of the Art Anwendungen konzentriert. Nachdem erfolgreichen auffinden, von mehreren Möglichkeiten zur Problembewältigung, ist der nächste Schritt das Erstellen eines Datensatzes. Hier muss beachtete werden, dass die Filterung aus dem originalen Datensatz, genug wichtige Features zur Klassifizierung mitbringt. Dazu wird der originale Datensatz auf die wichtigsten Inhalte überprüft und ein neuer Datensatz erstellt, in dem die Features innerhalb einer Zeile kombiniert werden.

Darauf folgend werden unterschiedliche Baselines anhand von mehreren State of the Art Lösungen produziert, an dem zukünftige Ergebnisse verglichen werden. Im Anschluss werden empirische Tests durchgeführt, welche eine potenzielle Lösung hervorbringen sollen. Anhand dieser Lösung sollen erste Ergebnisse begutachtet und Verbesserungsmöglichkeiten abgewogen werden. Nach der Implementierung der Verbesserungen soll eine Gegenüberstellung zum vorherigen Modell stattfinden und die Ergebnisse evaluiert werden. Die resultierenden Ergebnisse werden letztendlich im Kapitel 7.4 kritisch betrachtet.

Das Unterkapitel *Ausblick auf zukünftige Arbeiten* im Kapitel 8.3, dient dazu diese Arbeit abzurunden. Es soll den Leser Ideen und Anregungen für eigene Forschungsfragen vermitteln. Hierzu werden Ideen ausgearbeitet, an die der Leser anknüpfen kann.

1.3 Zielsetzung der Arbeit

Die Zielsetzung für diese Arbeit besteht darin, eine Verbesserung zu der aktuellen Software zu finden. Dabei sollen State of the Art Techniken ausprobiert werden, um eine Baseline zu schaffen. Im Folgenden sollen neue Lösungsansätze gesucht und ausprobiert werden. Dabei sollen hierarchische Ansätze priorisiert werden. Weiterhin soll eine hierarchische Klassifizierung angestrebt werden, um die Eigenschaften des Datensatzes und des DDC Systems zu

nutzen. Aufbauend darauf soll die Fragestellung geklärt werden, ob eine Verbesserung der hierarchischen Klassifikation durch Kombination der Vorhersagen der Hierarchieebenen erreicht werden kann? Am Ende werden die Ergebnisse analysiert und es soll ein Fazit daraus gezogen werden. Dabei soll die Frage beantwortet werden, ob eine Hierarchie zur Klassifizierung überhaupt einen Vorteil bringt. Zu guter Letzt werden aufbauende Arbeiten beschrieben, an den sich kommende Thesisarbeiten orientieren können.

2 Grundlagen

In diesem Kapitel soll ein grundlegendes Verständnis für die verwendeten Technologien aufgebaut werden. Weiterhin werden die wichtigsten Begriffe erklärt und eine Einleitung in die Thematik: *Verarbeitung von natürlicher Sprache* gestaltet. Ziel ist es, dem Leser die Technologien zu beschreiben, um das Verfolgen der Arbeit zu vereinfachen.

2.1 DDC System

Im Jahr 1873 entwickelte Melvil Dewey eine Universalklassifikation, die im Jahr 1876 ihr Debüt feierte. Diese Universalklassifikation ist als Dewey-Dezimalklassifikation (DDC), weitergehend auch als DDC System bezeichnet, bekannt geworden und ist international am erfolgreichsten unter den Klassifikationssystemen verbreitet. Das DDC System ist überaus flexibel ausgelegt und konnte somit auf wissenschaftliche Veränderungen über die Jahre hinweg erfolgreich reagieren [9]. Dabei soll das DDC System dazu dienen, archivierte und neu ankommende Objekte der DNB, intellektuell in eine Sachgruppe zu klassifizieren. Das DDC System setze sich bei der DNB im Jahr 2007 durch und wird für die Erschließung aller Dokumente genutzt [10].

Tabelle 1: DDC Hauptsachgruppen

000	Informatik, Informationswissenschaft & allgemeine Werke
100	Philosophie & Psychologie
200	Religion
300	Sozialwissenschaften
400	Sprache
500	Naturwissenschaften
600	Technik
700	Künste & Unterhaltung
800	Literatur
900	Geschichte & Geografie

Quelle: [11]

Die Tabelle 1 demonstriert, wie die zehn Hauptsachgruppen verteilt sind. Die aufgeführten Kategorien sind weiterhin in Unterkategorien aufgeteilt. In der zweiten Ebene ist eine Aufteilung bis zu 100 Sachgruppen möglich und in der dritten Ebene bis zu 1000 Sachgruppen. Dazu muss erwähnt werden, dass momentan nicht alle der 1000 möglichen Sachgruppen belegt sind. Das DDC System ist das meist genutzte Klassifikationssystem der Welt. In mehr als 135 Ländern wird dieses System in den Bibliotheken verwendet [11].

2.2 Natural Language Processing

Der Forschungsbereich Natural Language Processing (NLP) wird dazu genutzt, Maschinen beizubringen wie sie die menschliche Sprache und Schrift interpretieren können. Dafür werden Tools und Techniken entwickelt, um diese für bestimmte Aufgabenfelder zu nutzen. Aufgabenfelder, in denen NLP eingesetzt wird, sind zum Beispiel maschinelle Übersetzung, Text Verarbeitung und Spracherkennung. Bei der Erstellung einer NLP Applikation steht der Entwickler immer vor drei grundsätzlichen Hauptproblemen. Das erste Problem ist das Verarbeiten der gegebenen Daten. Das zweite Problem befasst sich mit der Repräsentation und der Bedeutung des gegebenen Inputs. Beim dritten Problem handelt es sich um die Kenntnisse der Wörter. Konkret geht es darum, dass ein NLP System sich Stück für Stück nach außen arbeitet. Zunächst werden die einzelnen Wörter und die dazu gehörige morphologische Struktur der Wörter betrachtet. Danach geht es auf die Satzebene, in der die Reihenfolge der Wörter betrachtet wird. Hinzukommen grammatikalische Aspekte und die Bedeutung des Satzes, sowie in welchem Kontext dieser steht [12]. Dies soll an dem Beispiel *Computer arbeitet nicht ausschalten* verdeutlicht werden. Hier ist die Satztrennung oder auch die Betonung des Satzes wichtig. Dieser kann auf zwei weisen interpretiert werden:

- Computer arbeitet nicht / ausschalten
- Computer arbeitet / nicht ausschalten

Selbst dem menschlichen Leser können hier Fehler passieren, wenn nicht der Kontext des Satzes bekannt oder die Zeichensetzung irreführend ist. Eine weitere Problematik besteht mit Homonymen, die für eine Maschine nicht so einfach interpretierbar sind. Die Bank ist eine Sitzgelegenheit und kann ebenso

ein Geldinstitut sein [13]. Ein anderes Beispiel wäre der Absatz, welcher am Schuh befestigt ist oder es ist der Textumbruch damit gemeint. Nach [14] ist NLP eine Reihe von Techniken zur Analyse und Darstellung vorkommender Texte. Um eine menschenähnliche Sprachverarbeitung zu imitieren, müssen eine oder mehrere Ebenen von den folgenden aufgeführten Ebenen erreicht werden:

- Phonologisch: Interpretation von Sprachlauten innerhalb und zwischen Wörtern
- Morphologisch: Komponentenanalyse von Wörtern, einschließlich Präfixe, Suffixe und Wurzeln
- Lexikalisch: Analyse auf Wortebene einschließlich lexikalischer Bedeutung und Sprachteil-Analyse
- Syntaktik: Analyse von Wörtern in einem Satz, um die grammatische Struktur des Satzes aufzudecken
- Semantisch: Bestimmen der möglichen Bedeutungen eines Satzes, einschließlich der Begriffsklärung von Wörtern im Kontext
- Diskurs: Interpretation von Strukturen und Bedeutungen von Texten, die größer sind als ein Satz
- Pragmatisch: Verständnis für den zielgerichteten Gebrauch von Sprache in Situationen, insbesondere für jene Aspekte von Sprache, die Weltwissen erfordern

Die aufgelisteten Ebenen sind applikations- oder aufgabenbezogen und somit nicht vollständig für jede Situation übertragbar [14].

Als Nächstes soll ein möglicher Textverarbeitungsprozess beschrieben werden. Der Verarbeitungsprozess beginnt mit einer morphologischen Analyse. Im Anschluss wird ein Stemming Verfahren in dem Query, als auch in den Dokumenten genutzt, um die Varianten eines Wortes zu reduzieren. Darauf aufbauend wird die lexikalische und syntaktische Bearbeitung angesteuert, die wiederum Lexika verwendet, um die Eigenschaften der Wörter zu bestimmen, Sätze zu analysieren und deren Sprachteile zu erkennen [12]. Wie oben erwähnt, sind nicht alle Ebenen nötig um einen Textverarbeitungsprozess durchzuführen.

2.3 Lernen mit Machine Learning Systemen

Machine Learning Systeme haben die Möglichkeit, über vier Arten trainiert zu werden, dabei gibt die Art des Lernens das System vor. Zur Wahl stehen hierfür:

- Überwachtes Lernen (Supervised Learning)
- Unüberwachtes Lernen (Unsupervised Learning)
- Halbüberwachtes lernen (Semi-supervised Learning)
- Verstärkendes Lernen (Reinforcement Learning)

Zunächst soll ein Überblick über das Überwachte lernen gegeben werden, gefolgt von den drei übrigen Methoden. Dabei werden typische Aufgaben und mögliche Machine Learning Systeme aufgezeigt [15].

2.3.1 Überwachtes Lernen

Die Klassifikation gehört zu den Standardaufgaben des überwachten Lernens. Dabei werden Instanzen, mit der dazugehörigen Klasse, aus dem Datensatz genommen und zum Lernen verwendet. Labels sind die Klassen eines Datensatzes beziehungsweise einer Instanz und genau diese, werden dem Algorithmus beim überwachten lernen übergeben. Ein Standard ist die Klassifizierung von E-Mails nach Spam oder Ham. Ein weiteres Aufgabenfeld des überwachten Lernens ist die Regression. Dabei wird als Beispiel der Preis eines Fahrzeugs anhand von Features wie Marke, Alter, Kilometerstand vorhergesagt. Geeignete Machine Learning Systeme für diese Aufgaben wären hierbei SVMs, lineare Regression, k-nearest neighbors oder logistische Regression [15].

2.3.2 Unüberwachtes Lernen

Das unüberwachte Lernen wird seinem Namen treu und versucht selbstständig, anhand des Datensatzes zu lernen. Im Datensatz gibt es keine Labels und der Algorithmus muss ohne Anleitung, geeignete Features finden. Die gefundenen Features aus dem Datensatz dienen dem Algorithmus zum Lernen. Hier können die Algorithmen in drei Kategorien unterteilt werden:

- Clustering
- Visualisierung und Dimensionsreduktion
- Lernen anhand Assoziationsregeln

Als mögliches Beispiel wird ein Studiengang genommen, von dem die Daten der StudentInnen vorliegen. Nun muss der Algorithmus selbstständig Verbindungen im Datensatz herausfinden. Als mögliches Ergebnis könnte, hervorkommen, dass 60 % der Besucher weiblich oder das 25 % der StudentInnen unter 22 Jahre alt sind. Untergruppen aus diesem Datensatz könnten mit hierarchischen Clusterverfahren erlernt werden [15].

2.3.3 Halbüberwachtes Lernen

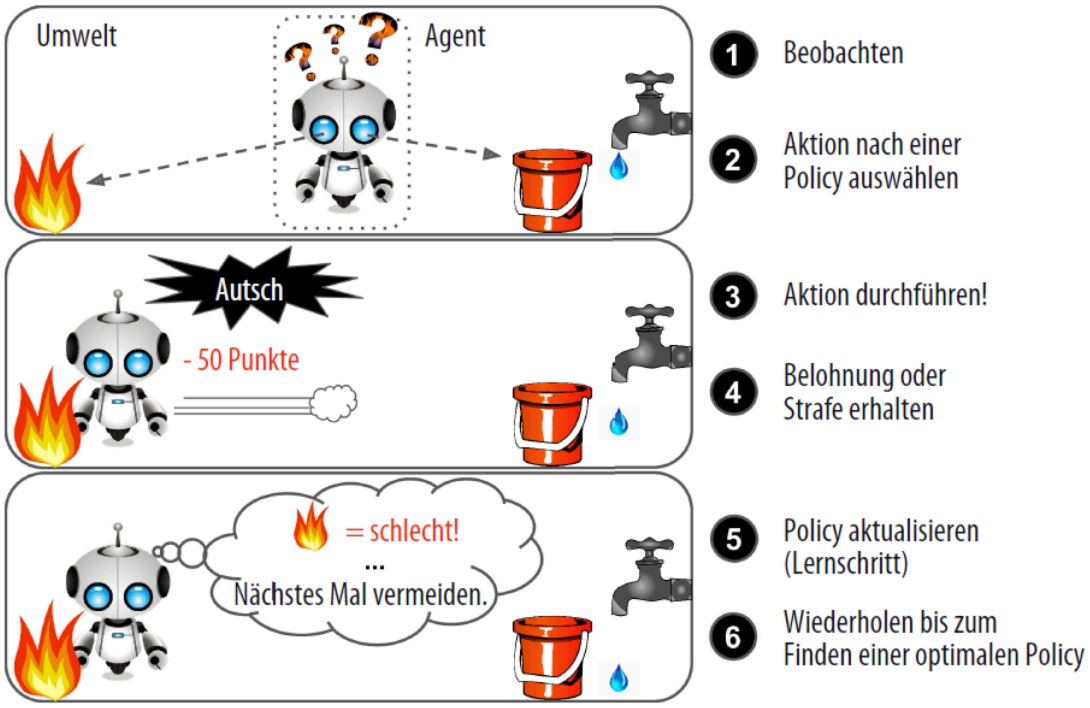
Diese Lernmethode wird beim Erkennen von Personen auf Bildern verwendet. Sehr bekannt ist diese Option von Facebook oder Google Photos. Dabei werden alle Bilder nach Personen durchsucht. Der Algorithmus erkennt dann, dass die Person A auf den Bildern 1, 44, 65 und die Person B auf 23, 44 zu sehen ist. Sobald Person A oder B mit einem Namen gelabelt wurde, wird der Name automatisch auf die Bereiche in den Bildern verteilt.

Die dafür vorgesehenen Algorithmen sind hierbei eine Mixtur aus überwachten und unüberwachten Lernmethoden. Für diese Methode könnten Deep Beliefs Networks (DBN) verwendet werden. Dabei besteht DBN aus mehreren hintereinander geschalteten restricted Boltzmann Machines (RBM). Diese werden nacheinander unüberwacht trainiert. Nach dem Durchlauf wird ein überwachtes Lernen zur Feinabstimmung des Systems verwendet [15].

2.3.4 Verstärkendes Lernen

Diese Lernmethode ist eine ganz besondere Art von Lernen und hat kaum was mit den oben genannten Arten zu tun. Der Algorithmus wird dabei als Agent angesehen und ist für die Entscheidungen innerhalb des Lernprozesses zuständig. Dies wird durch Beobachtung und darauf beruhende Aktionen erzielt. Durch Aktionen kann der Agent sich Belohnungen oder Strafen in einem festgelegten Zeitrahmen verdienen. Der Agent versucht dabei, die höchste Punktzahl zu erreichen. Dabei definiert die Policy die Aktionen, die zu bestimmten Situationen gehört.

Abbildung 1: Agent Entscheidungen



Quelle: [15]

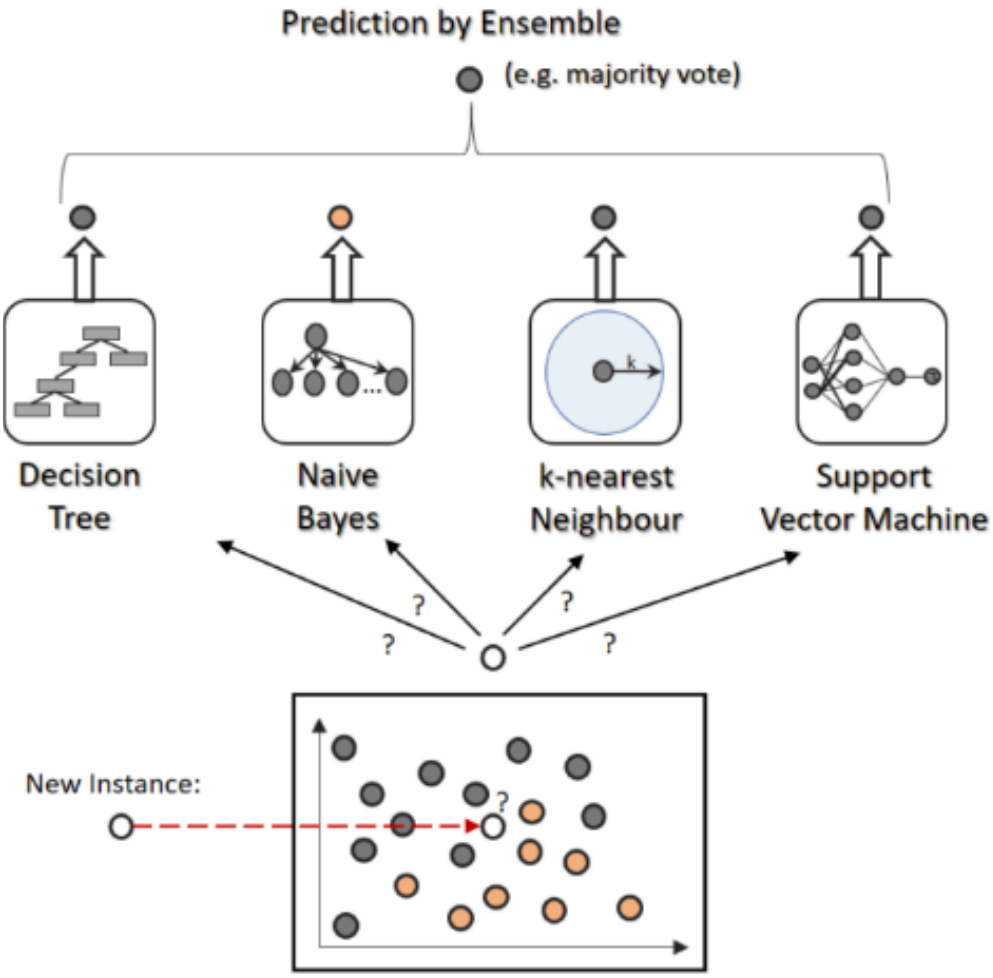
Verstärkendes Lernen kam 2017 in die Zeitungen, als die Maschine AlphaGo von DeepMind den Weltmeister Ke Jie im bekannten Brettspiel Go besiegte [15].

2.4 Ensemble Learning

Bei relevanten oder komplexen Problemstellungen sollte die Überlegung getroffen werden, ob es sinnvoll ist, sich nur auf einen Lernalgorithmus zu verlassen. Durch die Nutzung von unterschiedlichen Lernalgorithmen können auch weiterhin deren Parameter, sich situationsbedingt auf die Daten oder die Klassifizierung auswirken. Lernalgorithmen verhalten sich dabei unterschiedlich im Bereich der Über- und Unteranpassung der Modelle. Beim Ensemble Learning wird ein Ensemble von Lernalgorithmen oder auch Klassifizierer genannt gebildet, um ein Ensemble Average (Kollektivmittelwert) zu erstellen. Das bedeutet, alle Klassifizierer geben an, zu welcher Klasse eine Instanz gehört. Durch diese Art der Klassifizierung können einzelne Ausreißer durch Klassifizierer abgedämpft werden. Daher besteht die Annahme, dass durch mehrere unterschiedliche Algorithmen, ein besseres Ergebnis im Durchschnitt

erzeugt wird [16]. In der Abbildung 2 wird dieser Vorgang anhand des Hinzufügens, einer neuen Instanz illustriert. Als Lernalgorithmen wird dabei die Support Vector Machine, k-nearest Neighbour, Naive Bayes und ein Decision Tree verwendet. Diese vier Lernalgorithmen entscheiden durch das majority vote Verfahren, zu welcher Klasse die neue Instanz gehört.

Abbildung 2: Ensemble Learning



Quelle: [16]

Wie oben im Text aufgegriffen, ist das majority vote Verfahren ein recht simples aber nützliches Instrument zur Klassifizierung. Dies wird durch einen Voting Klassifizierer zustande gebracht, der einen Kollektivmittelwert bildet. Mithilfe von Ensemble Learning kann die Frage nach dem richtigen Klassifizierer umgangen werden, in dem unterschiedliche oder gleiche Lernalgorithmen mit verschieden Parameter simultan genutzt werden. Ziel dabei ist es, die Schwä-

chen einzelner Klassifizierer auszugleichen. Der Vorteil von Ensemble Learning ist dabei, das ein Ensemble mit vielen Klassifizierer oft ein besseres Ergebnis erzielt, als der beste einzelne Klassifizierer. Für Ensemble Learning gibt es drei unterschiedliche Ansätze. Diese drei Ansätze wären dabei:

- Bagging / Pasting (Gleicher Algorithmus aber unterschiedliche Daten)
- Stacking (Erweiterung des majority vote durch einen Meta Klassifizierer)
- Boosting (Sequentielles Ensemble Learning)

2.5 Wahrheitsmatrix

In diesem Teil soll die Wahrheitsmatrix oder auch im englischen *confusion matrix* erläutert werden. Der einfachheitshalber wird die confusion matrix an einer binären Klassifikation vorgestellt. Diese kann vier Zustände erreichen:

- TP (richtig positiv)
- FP (falsch positiv)
- FN (falsch negativ)
- TN (richtig negativ)

Diese Zustände entstehen durch die Kombination aus Realität und Klassifizierungsergebnis. TP stellt dar, dass die Klassifizierung richtig war und TN, falls der Klassifikator eine falsche Antwort liefert. Somit stehen die korrekten Klassifizierungen in einer Diagonalen, die in der Tabelle 2 grün dargestellt sind. Bei den Zuständen FP und FN ist es etwas verzwickter. FP kann anhand eines Diabetestests demonstriert werden. Falls der Test anzeigt, dass der Patient Diabetes erkrankt ist, jedoch in Realität gesund ist, wird das als FP eingeordnet. Bei FN wäre das der umgekehrte Fall. Wenn der Test kein Diabetes anzeigt, aber der Patient trotzdem daran erkrankt ist [17].

Tabelle 2: Confusion Matrix

		Realität	
		Ja	**Nein**
Entscheidung des Klassifikator	**Ja**	TP (richtig positiv)	FP (falsch positiv)
	Nein	FN (falsch negativ)	TN (richtig negativ)

Quelle: Eigene Darstellung

Die confusion martrix dient weiterhin dazu aufzuzeigen, wie gut der Klassifizierer arbeitet. Im nächsten Teil werden vier Berechnungen vorgestellt und wie diese anzuwenden sind. Precision und Recall werden dabei klassenabhängig betratet

2.5.1 Precision

Die Precision gibt den Anteil der richtigen Ergebnisse über alle positiven Ergebnisse wieder. Dabei wird die Precision mit der Formel *TP / (TP+FN)* berechnet.

2.5.2 Recall

Der Recall gibt den Anteil aller korrekten Ergebnisse wieder. Ein Beispiel könnte hier sein, alle Patente durchzusehen und jene wiederzugeben, die auch richtig sind. Die Formel hierbei lautet *TP / (TP+FP).*

2.5.3 F-Measure

Der F-Mearsure oder auch das harmonische Mittel genannt, wird mit der Formel *2*(Precision * Recall) / (Precision + Recall)* berechnet. Das harmonische Mittel ist für ungleichmäßig verteilte Datensätze geeignet.

2.5.4 Accuracy

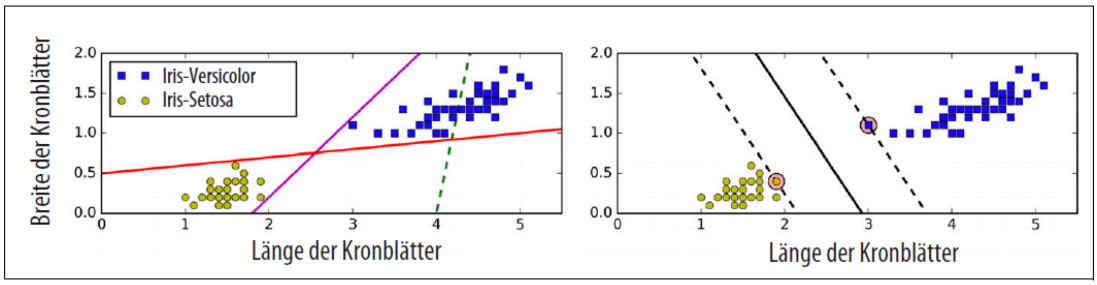
Die Genauigkeit wird mit der Formel *(TP+TN) / (TP+FP+FN+TN)* ausgerechnet. Die Accuracy eignet sich für ausgewogene Datensätze.

2.6 Support Vector Machine

Die Support Vector Machine (SVM) kann zur Bewältigung von linearen und nicht linearen Problemstellungen verwendet werden. Dabei wird sie zur Klassifizierung, Regression oder zur Anomalie Erkennung eingesetzt. Dabei gehört

die SVM zu den flexiblen und mächtigen Machine Learning Modellen [15]. Anhand der Abbildung 3 soll die Funktionsweise der SVM demonstriert werden. Die Abbildung 3 wurde mit dem gut bekannten Iris-Datensatz trainiert und die Ergebnisse visualisiert. Es soll dabei eine klare Entscheidungsgrenze zwischen den beiden Kategorien erstellt werden. Die linke Hälfte der Abbildung 3 wurde mit drei unterschiedlichen linearen Klassifikatoren trainiert und die rechte Hälfte mit einem SVM-Klassifikator. Im Vorfeld sollen die Ergebnisse auf der linken Hälfte erklärt werden. Die gestrichelte grüne Linie ist so schlecht trainiert, dass der Klassifikator die zwei Klassen nicht voneinander unterscheiden kann. Die gestrichelte grüne Linie geht direkt durch den Datensatz und ist somit nicht zu gebrauchen. Die beiden durchgezogenen Linien schaffen es zwar die Klassen voneinander zu trennen, liegen jedoch nah an den Dateneckpunkten. Das hätte zur Folge, das neu eingespielte Daten eventuell falsch klassifiziert werden. Somit wurde eine saubere Trennung anhand der Trainingsdaten erstellt, jedoch könnten neue Datenpunkte falsch zugeordnet werden. Bei dem rechten Teil der Abbildung 3 Large-Margin-Klassifikation wird aufgezeigt, wie ein SVM-Klassifikator optimalerweise funktionieren soll. Bei der Trennung der zwei Klassen wird eine sogenannte Straße aufgebaut, die Large-Margin-Klassifikation. Dabei hält diese Straße den größtmöglichen Abstand zwischen den Trainingsdatenpunkten. Somit können problemlos neu Datenpunkte hinzugefügt werden, welche die Entscheidungsgrenze nicht beeinflussen.

Abbildung 3: Large-Margin-Klassifikation

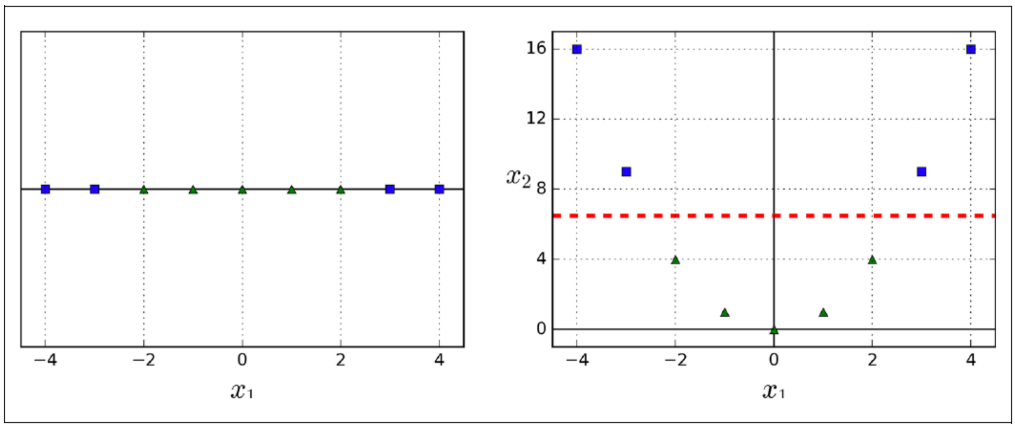


Quelle: [15]

Wie erwähnt lassen sich SVM zur Klassifizierung von nicht linearen Datensätzen verwenden. An der Abbildung 4 wird aufgezeigt, wie sich das Diagramm optisch verändert und dass die Datenpunkte linear separierbar geworden sind.

Um den Datensatz linear separieren zu können, werden weitere Merkmale benötigt. In diesem Fall kamen dafür polynomiale Merkmale zum Einsatz. Einfach ausgedrückt, der vorherige Datensatz besitze nur ein Merkmal. Dieses Merkmal wird als $\mathbf{x}_1$ beschrieben. Beim Hinzufügen eines zweiten Merkmals $\mathbf{x}_2 = (\mathbf{x}_1)^2$ kann die gewünschte Krümmung und ein zweidimensionaler Datensatz erzeugt werden [15]. Diese Lösung kann auch auf Probleme in höheren Dimensionen reproduziert werden und ist somit ein guter Ansatz für höher dimensionale Problemstellungen.

Abbildung 4: Transformation eines nichtlinearen Datensatzes



Quelle: [15]

Abschließend kann gesagt werden, dass SVMs effizient und gut anpassbar auf verschiedene Anwendungsfälle sind.

2.7 Naive Bayes Klassifikation

Um eine Klassifikation auf Berechnung bedingter Wahrscheinlichkeiten durchzuführen, wird die Naive Bayes Klassifikation herangezogen. Dazu wird der Satz von Bayes verwendet, welcher seinen Ursprung in der Mathematik findet und zu der Wahrscheinlichkeitstheorie gehört.

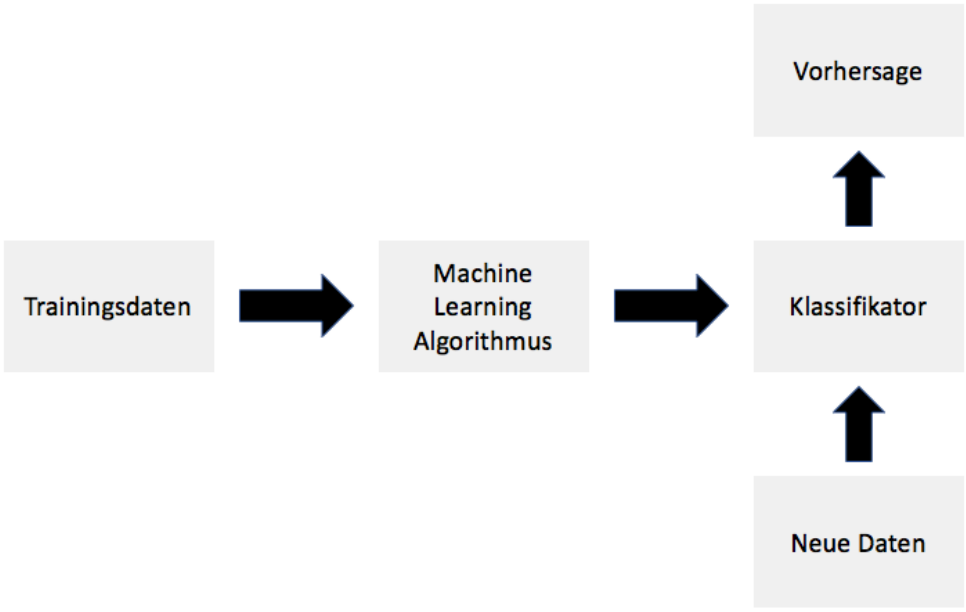
Abbildung 5: Satz von Bayes

$$P(A_i|B) = \frac{P(A_i \cap B)}{P(B)}$$

Quelle: [18]

Naive Bayes Klassifikatoren werden beispielsweise für supervised classification genutzt. Der Naive Bayes Klassifikator berechnet die Wahrscheinlichkeit, zu welchem Label eine neue dazukommende Instanz gehört. Ein klassisches Szenario ist die Klassifizierung von E-Mails in Spam und nicht Spam Nachrichten. Dabei lässt sich erkennen, dass es sich hierbei um einen linearen Klassifikator handelt. Durch den simplen aber effizienten Aufbau der Naive Bayes Klassifikation, lassen sich sehr gut funktionierende Modelle erstellen, die sich wiederum gut für Text- oder Dokumentklassifikation verwenden lassen [19]. Dazu kommt, dass der Naive Bayes Klassifikator leicht zu implementieren ist, eine gewisse Robustheit aufweist und präzise ist, was die möglichen Einsatzgebiete erweitert [19]. Das Wort *naive* soll hierbei die Annahme hervorbringen, dass die Features aus dem Datensatze unabhängig voneinander sind, auch wenn dies in der Praxis nicht immer der Fall ist [20].

Abbildung 6: Standardablauf eines Klassifikationsmusters



Quelle: Eigene Darstellung

Um den Naive Bayes Klassifikator für Text- oder Dokumentenklassifizierung zu verwenden, müssen gewisse Vorbereitungen getroffen werden. Eine Aufgabe davon ist Feature Extrahierung und Selektion. Dabei sind drei Punkte zu beachten [19]:

- Hervorstechend. Die Features sind wichtig und bedeutsam
- Invariant. Bezeichnet die Unveränderbarkeit von Größen. Die Features werden bei wechselnden Bedingungen nicht verändert. Wichtig im Kontext für Bildklassifizierungen.
- Diskriminierend. Features beinhalten genügend Informationen, um zwischen Mustern zu unterscheiden.

Um die Daten für das Trainieren von Text vorzubereiten, wird die Technik *Bag of Word* aus dem NLP Repertoire verwendet. Danach wird eine Vektorisierung der Wörter vorgenommen. Dabei ist es entscheidend, ob das Multinominale- oder das Bernoulli Model verwendet wird. Beim Bernoulli Model werden binäre Daten verlangt. Das bedeutet, dass jedes Token in dem Feature Vektor eines Dokuments mit **1** oder **0** repräsentiert wird. Der Feature-Vektor besitzt m-Dimensionen, dabei stellt m die Anzahl der Wörter im Vokabular dar. Die Repräsentation durch **1**, bedeutet, dass ein gewisses Word in dem Dokument vorhanden ist. Beim multinominalen Model wird hingegen ein Produkt aus der Term Frequency und der Inverse Document Frequency gebildet. Dieses Produkt wird für die anstehende Berechnung verwendet [19].

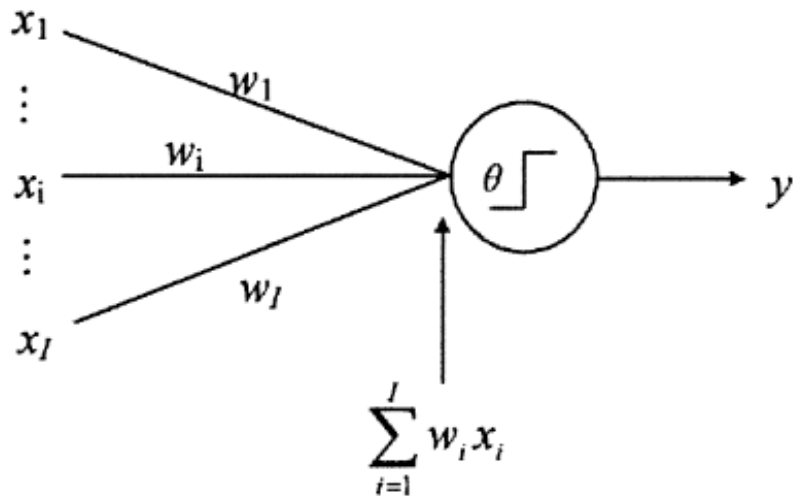
Empirische Ergebnisse belegen, dass die Performance bei Multinominalen Model mit einem großen Vokabular wesentlich effizienter ist. Jedoch bestimmt die Wahl der Features als auch die dazugehörige Verarbeitung wesentlich die resultierenden Ergebnisse [21]. Zugrunde liegt, dass es sich lohnt, Tests mit unterschiedlichen Features und Vorverarbeitungen zu gestalten. Danach sollten die Ergebnisse miteinander verglichen werden [19].

2.8 Künstliche neuronale Netze

In dem folgenden Unterkapitel sollen unterschiedliche neuronale Netze beschrieben werden. Dabei soll die Funktionsweise aufgeführt und ein fundamentales Wissen gebildet werden.

Grundlegend ist hier zu erwähnen, dass das neuronale Netz Ihre Ursprungsidee aus der Natur besitzt. Im Genauen ist dabei das menschliche Gehirn gemeint. Das Gehirn lernt, passt an, assoziiert und kann Informationen parallel verarbeiten. Diese Fähigkeiten können mit Maschinen nachgebildet werden, um komplexe technische Probleme zu lösen. Die künstlich nachgebildeten neuronalen Netze können mithilfe von Daten lernen und sich dabei selbst organisieren. Hierbei bauen die künstlichen Netze auf einzelne, jedoch gleiche Konnektoren auf. Dieser Konnektor heißt Neuron oder Perzeptron [22].

Abbildung 7: Einfaches Perzeptron



Quelle: [23]

Die Abbildung 7, stellt ein einfaches Perzeptron dar. Daran ist gut zu erkennen, dass das Perzeptron mehrere Eingänge mit $\mathbf{x_i}$ besitzen kann. Weiterhin ist jedes Eingangssignal mit einem Gewicht $\mathbf{w_i}$ versehen. Das Perzeptron besitze dazu noch einen Schwellenwert, der mit θ symbolisiert wird. Die Ausgabe wird mit γ gekennzeichnet [23]. Mit diesem Perzeptron lassen sich Boolesche Funktionen ausführen, dass soll an der booleschen AND Funktion demonstriert werden. Die Tabelle 3 soll mögliche Kombinationen des Eingangssignals aufführen und das dazugehörige Ergebnis in der letzten Zeile widergeben.

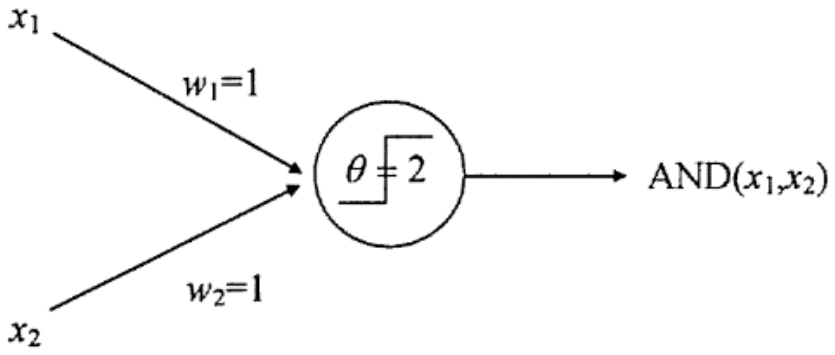
Tabelle 3: Boolesche AND Funktion

	Werte			
$\mathbf{x}_1$	0	0	1	1
$\mathbf{x}_2$	0	1	0	1
AND($\mathbf{x}_1$, $\mathbf{x}_2$) = $\mathbf{x}_1$ $\mathbf{x}_2$	0	0	0	1

Quelle: [23]

Diese Werte können in ein simples Perzeptron übernommen werden. Der Schwellwert kontrolliert in gewisser Weise die Ausgabe des Perzeptrons. Erst sobald dieser erreicht oder überschritten ist, findet eine Ausgabe statt.

Abbildung 8: Darstellung der AND Funktion an einem Perzeptron



Quelle: [23]

Dieses Perzeptron besitzt den Schwellwert 2 und dieser wird erst erreicht, wenn $\mathbf{x}_1$ und $\mathbf{x}_2$ jeweils eine **1** als Input mitgeben. Mit diesem Wissen lässt sich sagen, das ein neuronales Netz, ein Verbund von vielen Perzeptrons ist. Typische Anwendungsgebiete von neuronalen Netzen liegen in den Bereichen:

- Mustererkennung und Klassifikation
- Modellbildung, Identifikation, Prognose und Simulation
- Signalverarbeitung, Datenanalyse, Diagnose, Bewertungssysteme
- Steuerung, Regelung

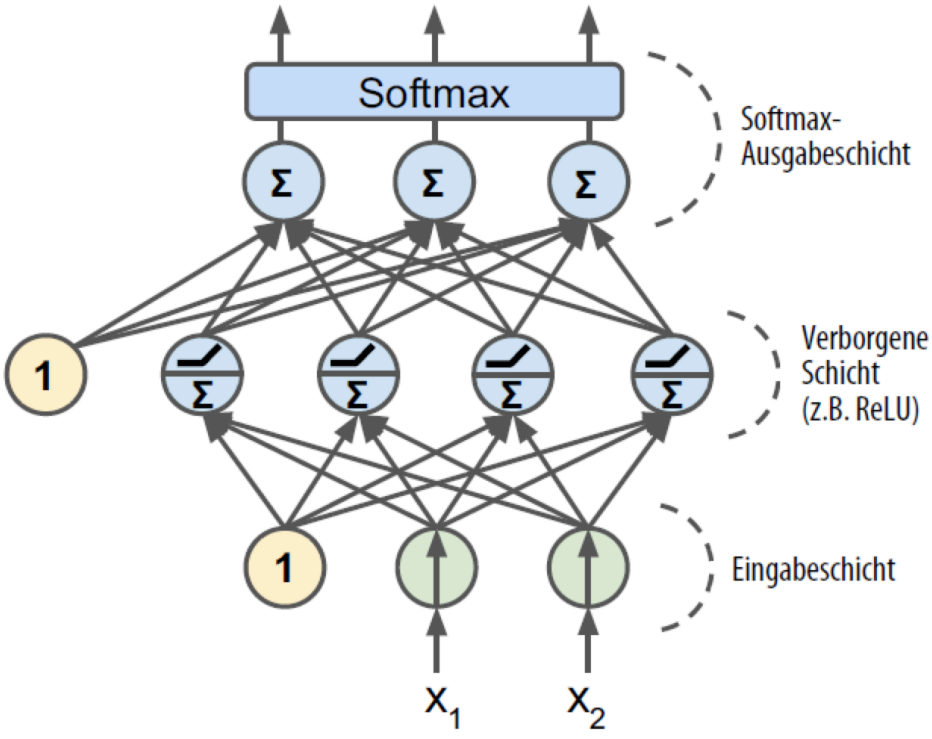
Um weiter in die Thematik einzudringen, muss der Begriff Deep Neural Network (DNN) erläutert werden. Dieser steht als Überbegriff für eine Ansammlung unterschiedlicher Algorithmen. Diese haben als Gemeinsamkeit, dass sie mehr als eine verdeckte Schicht, weitestgehend als Hidden Layer genannt, besitzen. Die Idee dahinter ist es, komplexe Problemstellungen mit tiefgehen-

den Schichten zu lösen. Mit einem einfachen Perzeptron lassen sich die booleschen Funktionen AND und OR ausführen. Um die XOR Funktion zu implementieren, sind mehrere Schichten nötig. Durch Nutzen eines Multilayer Perceptron (MLP), kann die Boolesche XOR Funktion implementiert werden [23].

2.8.1 Multilayer Perceptron

In der Abbildung 9 wird ein MLP aufgezeigt, welches einen Input Layer, einen Hidden Layer, Output Layer und als finalen Layer einen Softmax besitzt. Dabei sind alle Neuronen, bis auf den Output Layer, vollständig miteinander verbunden. Eine vollständig miteinander verbundene Schicht, wird auch als Dense Layer bezeichnet. Beim Hinzufügen eines weiteren Hidden Layers, würde das dadurch entstehende Netz zu der Kategorie *Deep Learning Netze* gehören. In der Ausgabeschicht wird die Softmax Aktivierungsfunktion dazu benutzt, einen realen Wert auszugeben. Wenn ein MLP über cross-entropy Kostenfunktion trainiert wird, kommen im Ausgabe Layer an dem Index ¡ Vektoren an. Diese werden über die Softmax Aktivierungsfunktion umgewandelt und sind dadurch unter anderem für Menschen nachvollziehbar.

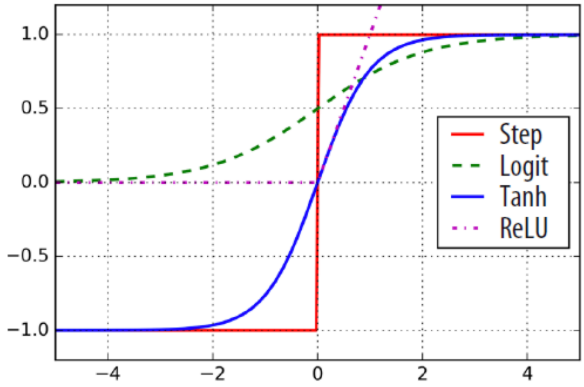
Abbildung 9: Multilayer Perceptron



Quelle: [15]

Die Kostenfunktion gibt die Differenz zwischen dem vorhergesagten Wert und dem tatsächlichen Wert wider [24]. Während des Lernvorgangs sollte dabei die Differenz optimalerweise abnehmen. Dabei wird eine Zahl im Wertebereich von **0** bis **1** ausgegeben. Der Ausgabewertwert sollte so klein wie möglich gehalten werden. In der Vergangenheit galt die Sigmoid Aktivierungsfunktion für lange Zeit als wissenschaftlicher Standard. Die Sigmoid Kurve ähnelt stark einem **S** und ist weiterhin eine Tanh Funktion. Die Sigmoid Aktivierungsfunktion wurde durch die ReLu Aktivierungsfunktion aufgrund von besseren und schnelleren Ergebnissen in einem neuronalen Netzwerk ersetzt.

Abbildung 10: Aktivierungsfunktionen



Quelle: [15]

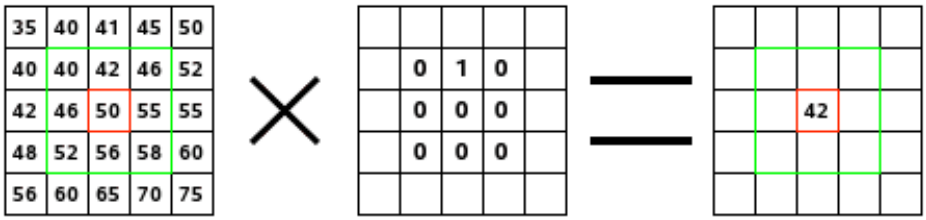
Zum Trainieren des Netzwerks wird der Backpropagation Algorithmus verwendet, welcher unter anderem als Gradientenverfahren mit Autodiff im Reverse-Modus beschrieben wird. Das Gradientenverfahren ist ein iteratives Optimierungsverfahren, bei dem pro Durchlauf die Modellparameter optimiert werden. Autodiff ist weiterhin als automatische Differenzierung bekannt. Dabei geht es um die Optimierungsknoten, denen es erlaubt ist, nach Parametern zur Minimierung der Kostenfunktion zu suchen [15].

2.8.2 Convolutional Neural Network

Dieses Netz ist durch seinen Aufbau, perfekt für Bilder- oder Videoerkennung geeignet. Mit steigender Rechenkapazität und Leistung der Maschinen kamen neue Aufgabenfelder hinzu. Eins dieser Felder ist die Verarbeitung oder Erkennung von Text. Doch zunächst sollen allgemein beschrieben werden, wie ein Convolutional Neural Network (CNN) funktioniert.

Die wichtigste Komponente, die ein CNN ausmacht, ist der Convolutional Layer. Der Convolutional Layer kann auch als Faltungsmatrix bezeichnet werden. Ziel einer Faltungsmatrix ist dabei, eine Bildverarbeitung hervorzubringen. Hierzu werden störende Anteile eines Bildes reduziert und wichtige Anteile hervorgehoben. In der Abbildung 11 wird auf der linken Seite das Eingabebild in Farbwerte dargestellt, auf die der Filter oder auch Kern genannt, gelegt wird. Der Kern ist hier eine 3x3 Matrix, da dieser Filter weit verbreitet ist. Der Filter geht hierbei Pixel für Pixel durch das Eingabebild und führt eine Multiplikation mit den 8 Nachbarpixeln durch. Das multiplizierte Ergebnis wird dabei als neuer Wert des aktuellen Pixels übernommen. Im rechten Teil des Bildes wird mit dem grünen Rand der Filter oder auch Kern repräsentiert und die rote Umrandung, stellt das aktuelle Pixel dar [25]. Der neue Wert des Pixels wurde durch die Multiplizierung von 40x0 + 42x1 + 46x0 + ... = 42 erzeugt.

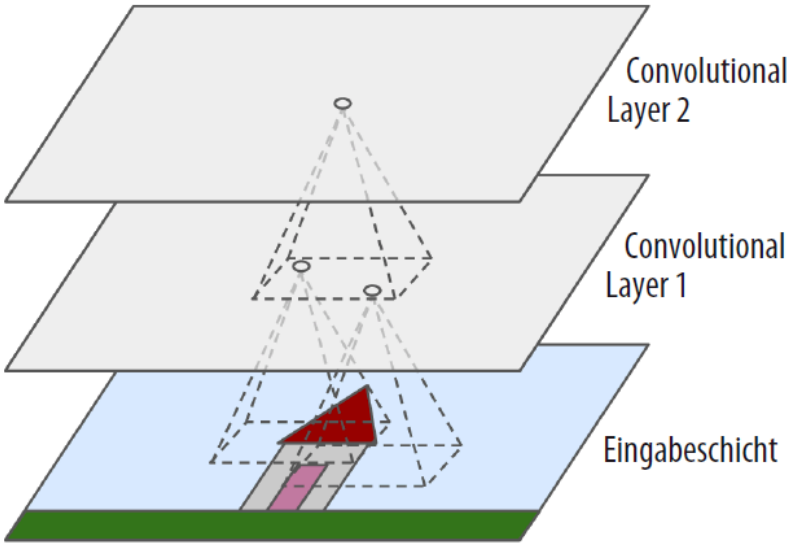
Abbildung 11: Anwendung eine Faltungsmatrix



Quelle: [25]

In der Abbildung 12 werden drei Layer aufgezeigt, dabei beinhalten die Convolutional Layer 1 und 2 Neuronen, die als Kreis abgebildet sind. Wie in der Abbildung 12 zu sehen ist, sind nicht alle Neuronen im Ersten Convolutional Layer, mit den Pixeln in der darunterliegenden Eingabeschicht verbunden. Das Wahrnehmungsfeld der einzelnen Neuronen bestimmt die zu betrachtenden Pixel. Dabei gilt dieses Prinzip auch für den zweiten Convolutional Layer. Neuronen aus dem zweiten Convolutional Layer, betrachten nur kleine Rechtecke aus dem darunterliegenden Layer. Durch Beibehalten dieses Stiels, kann der erste Convolutional Layer sich auf kleinteilige Merkmale in der Eingabeschicht fixieren. Im nächsten Layer werden die vorherigen Merkmale des ersten Convolutional Layers, zu übergeordnete Merkmale zusammengefasst. Durch die beschriebene hierarchische Struktur der Schichten, die für beispielsweise Bildern verwendet wird, kann das CNN gut funktionieren [15].

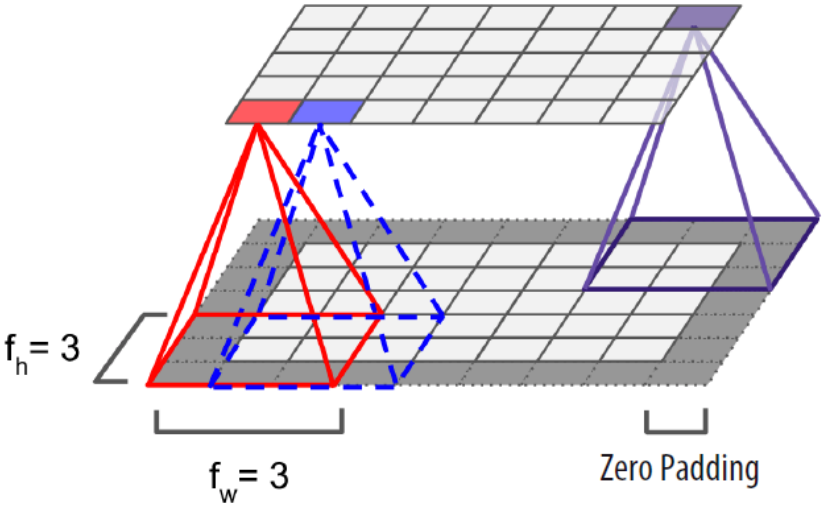
Abbildung 12: Schichten eines CNNs



Quelle: [15]

Die jetzt verwendeten Schichten formen die Daten in sogenannte 2-D-Daten beziehungsweise zu einer Matrix um. Die Neuronen lassen sich weiterhin zu ihren jeweiligen Eingaben oder der vorherigen Schicht zuordnen. Um das zu erläutern, wird ein Neuron aus der Zeile i und Spalte j genommen. Mochte man die dazugehörige verbundene Ausgabe verwenden, wird aus der vorherigen Schicht die Zeilen **i** bis $\boldsymbol{i + f_h - 1}$ und den Spalten **j** bis $\boldsymbol{j + f_w - 1}$ genommen. Die Höhe und Breite des Wahrnehmungsfelds werden durch die Werte $\mathbf{f_h}$ und $\mathbf{f_w}$ definiert. Um die gleichen Maße der vorherigen Schicht zu reproduzieren, werden um die Eingaben herum Nullen platziert. Dieser Vorgang wird als Zero Padding bezeichnet. Die Abbildung 13 soll dabei Zero Padding illustrieren. Dabei findet sich ein weiteres Merkmal aus Abbildung 13. Die Schrittweite des Wahrnehmungsfelds wird als stride bezeichnet [15].

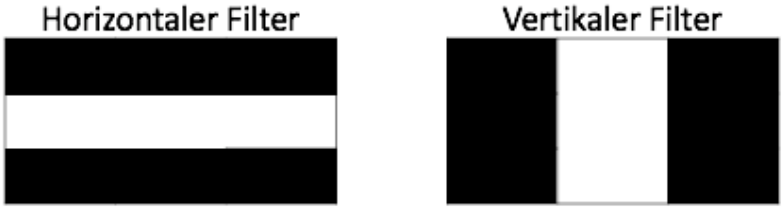
Abbildung 13: Zero Padding



Quelle: [15]

Für die Gewichtung der Neuronen in einem CNN gibt es zwei Ansätze, die verfolgt werden können. Diese zwei Ansätze sind unter dem Namen Filter oder Convolution Kernel bekannt. Die Höhe und Breite der Filter sind in diesem Beispiel an das Wahrnehmungsfeld von 3 x 3 angepasst. Bei beiden Filtern repräsentieren die schwarzen Felder eine **0** und die weißen Felder eine **1**. Sobald dem Neuron Gewichte hinzugefügt werden, beachtet das Wahrnehmungsfeld nur noch die weißen Felder [15].

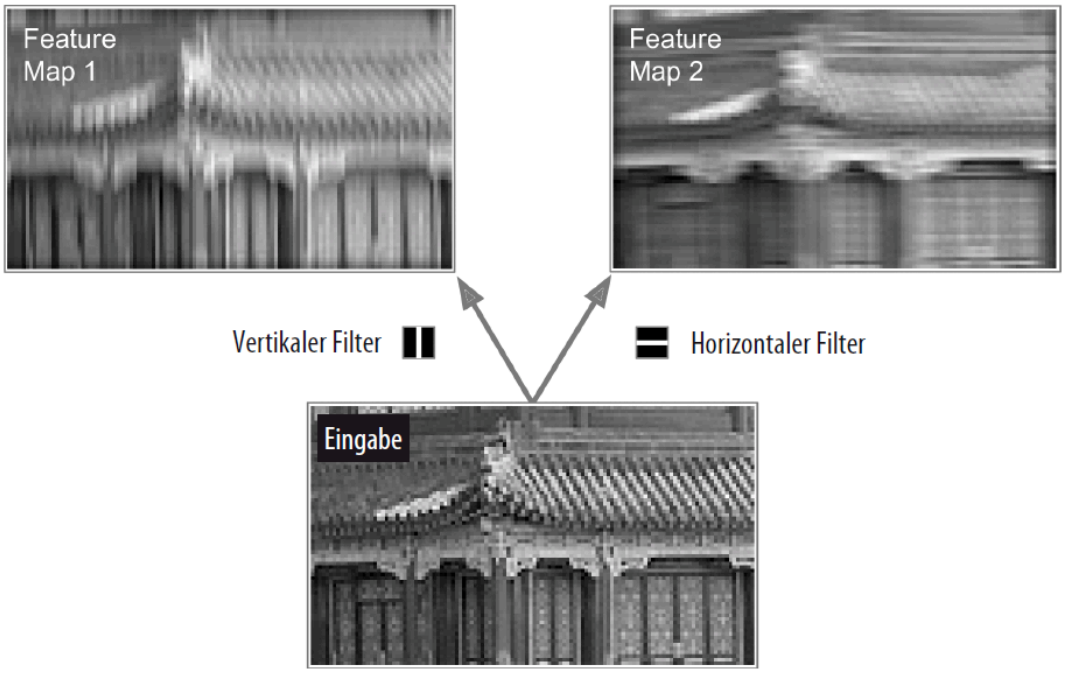
Abbildung 14: CNN Filter



Quelle: Eigene Darstellung

Im folgenden Bild wird aufgezeigt, wie sich die Filterung auf das Eingabebild ausübt. Bei der Verwendung des vertikalen Filters, werden die vertikalen weißen Linien verstärkt. Dabei verschwimmt der restliche Teil des Bildes. Beim horizontalen Filter wird genau das Gegenteil erzielt. Durch die Anwendung der Filterung über eine gesamte Schicht lässt sich ein Feature Map erstellen.

Abbildung 15: Anwendung von Filtern



Quelle: [15]

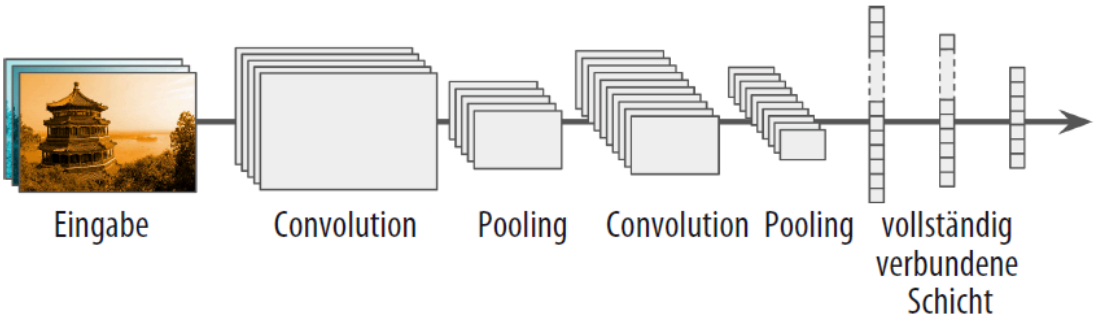
Die genannten Feature Maps werden beim Training eines CNNs erstellt, dabei werden automatisch die nützlichsten Filter erzeugt. Zudem wird erlernt, wie die Feature Maps zu komplexe Muster kombiniert werden. Diese Technik lässt sich Stapeln und ein Convolutional Layer kann mehrere Feature Maps besitzen, die zu einer 3-D-Schicht innerhalb des Layers wachsen [15].

Als Nächstes wird der Pooling Layer von CNN beschrieben. Dieser sollte mit dem jetzigen Wissen recht einfach zu verstehen sein. Dieser Layer verkleinert das Eingabebild mit einer sogenannten Unterstichprobe. Dabei werden die Parameter verkleinert und adäquat verringert sich die Rechenlast. Nebenbei wird somit Overfitting des CNNs vorgebeugt. Das Besondere an dem Pooling Layer ist, dass die Neuronen dieser Schicht nicht gewichtet werden. Der Pooling Layer besitzt jedoch Aggregationsfunktionen wie mean oder max, die benutzt werden, um die Eingabedaten zu aggregieren [15].

Im Anschluss soll eine typische Architektur eines CNN aufgeführt und in groben Zügen erklärt werden. Auf die Eingabe Schicht, folgen mehrere aufeinandergestapelte Convolutional Layer, aufbauend sollte auf jede Schicht eine ReLu- Schicht folgen. Als Nächstes folgt der Pooling Layer. Diese Reihenfolge

kann sich beliebig oft wiederholen. Dabei verringert sich das Eingabebild immer weiter beim Durchlaufen des Netzes, gewinnt jedoch an Tiefe. Am Ende dieser Ereigniskette befindet sich ein Feed-Forward-Netz mit einigen Dense Layers. Diese werden durch eine Softmax Ausgabeschicht vervollständigt [15].

Abbildung 16: Typische Architektur eines CNN



Quelle: [15]

Auf diesen Grundlagen soll nun die Verarbeitung von Texten oder Dokumenten erklärt werden. Dabei sind die wichtigsten Elemente bereits im oberen Teil beschrieben. Anhand der Abbildung 17 wird der Satz *I like this movie very much!* mit zwei möglichen Ausgaben klassifiziert. Dies wird in fünf Arbeitsschritten bewerkstelligt.

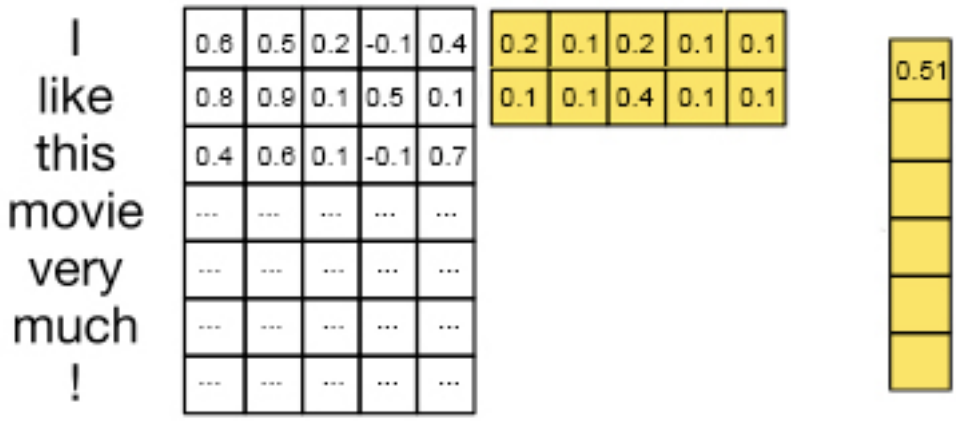
Schritt 1: Der eingelesene Satz mit sechs Wörtern und einem Sonderzeichen wird zunächst in eine fünfstellige Dimension umgewandelt. Eine Dimension beschreibt dabei den verwendeten Vektoren für ein Wort. Somit wurde eine Matrix mit dem Format ***s*** x **d** (7x5) angelegt.

Schritt 2: Texte besitzen lediglich eine Dimension, dennoch ist die Reihenfolge der Wörter in einem Text ausschlaggebend. Dieses Problem wird durch die Vektorisierung umgangen und eine Matrix für die 2-D Ansicht erstellt. Dabei wird ein Wort durch einen Vektor repräsentiert.

Schritt 3: Es wird ein zwei Wörter Filter basierend auf einer Matrix erstellt. Der Filter wird in der Abbildung 17 durch die gelbe Matrix illustriert. Dabei überlappt der Filter die Vektoren der Wörter *I* und *like*. Es wird nun ein Produkt aus den Elementen von *I* und *like* erstellt (0,6 x 0,2 + 0,5 x 0,1 + ... + 0,1 x 0,1 = 0,51). In den Feature Maps Layer wird danach das Produkt hinzugefügt. Der Filter bewegt sich nach der Berechnung eine Position nach unten und wird dabei,

für die Wörter *like* und *this* verwendet. Diese Prozedur wird bis zum letzten Wort beziehungsweise dem Sonderzeichen *!* fortgeführt.

Abbildung 17: CNN Convolution



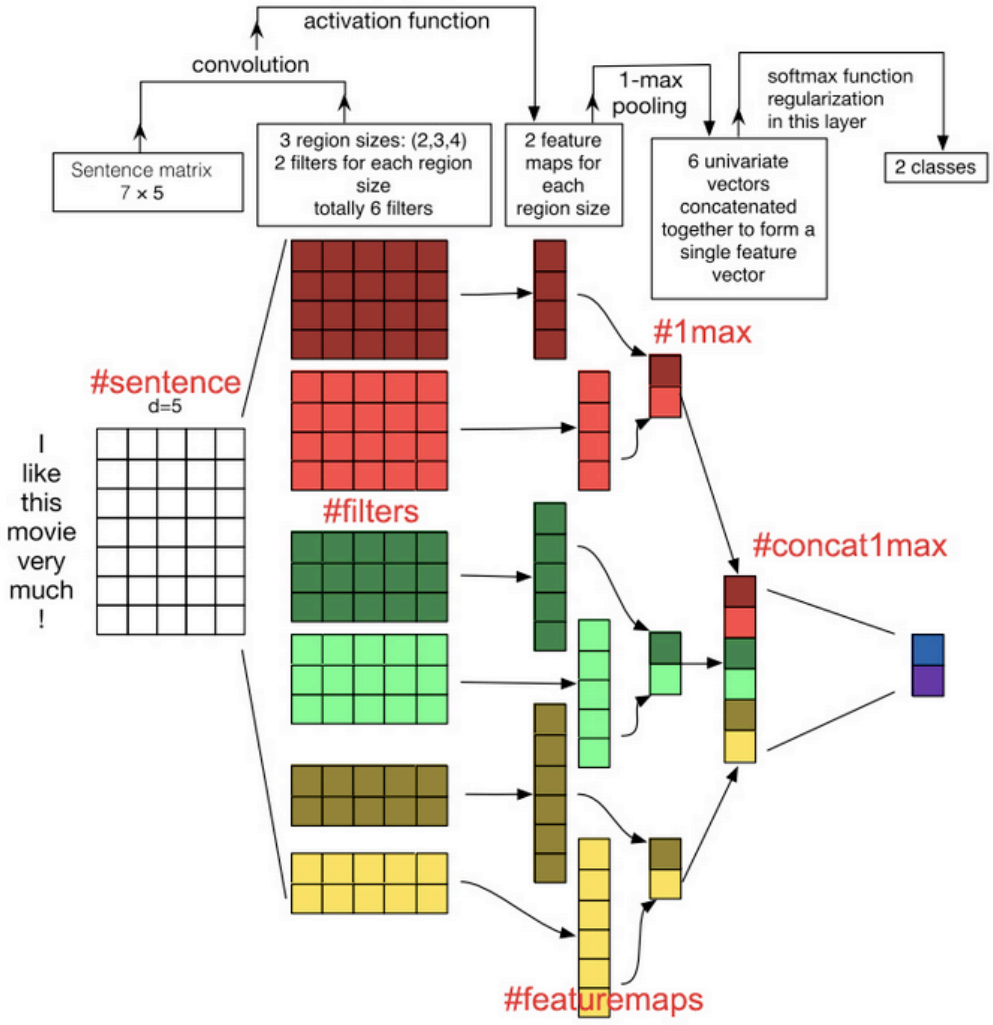
Quelle: [26]

Schritt 4: Es wird die Max-Pooling-Funktion auf dem Feature Maps Layer angewendet, um das größte Ergebnis zu bestimmen.

Schritt 5: Der zweitletzte Layer besitzt eine feste Länge von sechs aufgebauten Elementen. Diese beinhalten die besten sechs Ergebnisse, des vorherigen Layers. Die sechs Elemente werden nun dem finalen Softmax Layer übergeben, um die Klassifizierung durchzuführen. Weiterhin wird durch den Backpropagation-Algorithmus das CNN trainiert.

In der Abbildung 18 werden die soeben beschriebenen Schritte, in einem ganzen Prozess dargestellt. Dabei sollen die verwendeten Matrizen illustriert werden, um den beschriebenen Schritten einfacher folgen zu können [26].

Abbildung 18: Textklassifizierung anhand zwei Klassen



Quelle: [26]

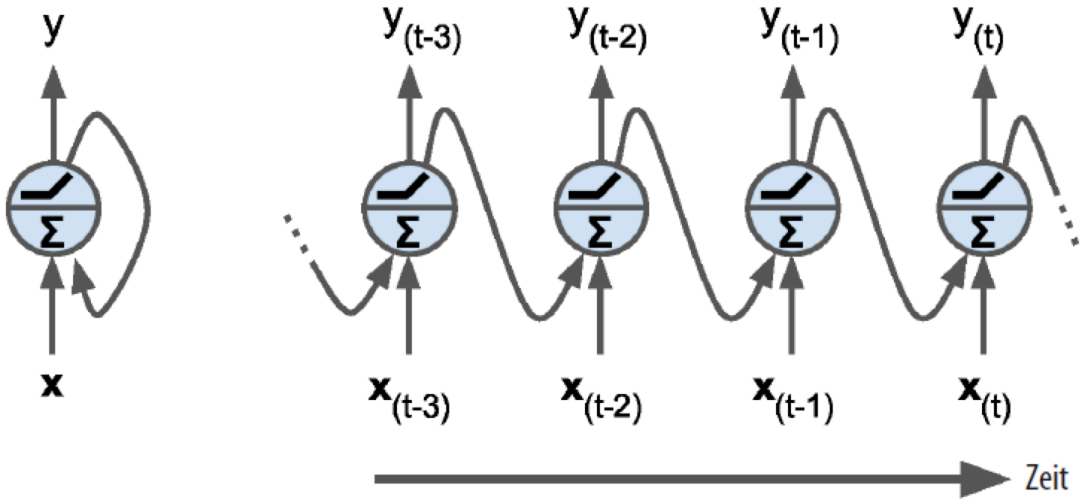
2.8.3 Recurrent Neural Network

Diese Art von Netz funktioniert sehr gut im Hinblick auf anhaltende Vorhersagen. Deshalb findet das Recurrent Neural Network (RNN) oft Verwendung an der Börse, um ohne Unterbrechung mögliche Aktienkurse vorherzusagen. Ein weiteres Einsatzgebiet ist in der Automobil Industrie. Dort wird das RNN im Bereich des autonomen Fahrens eingesetzt. Das RNN besitzt die Eigenschaft, sehr gut mit Sequenzen unbestimmter Länge umzugehen. Wie in Abbildung 19 angedeutet, kann sich das Netzwerk beziehungsweise das rekurrent Neuron über die Zeit hinweg selbstständig ausführen und somit dauerhaft mit

neuen Informationen versorgen. Somit lässt es sich sehr gut in Bereichen einsetzen, in denen dauerhaft neue Informationen anfallen. Deshalb ist das RNN sehr beliebt im NLP Bereich, um Sprache und Text zu analysieren [15].

Bei einem Feed-Forward-Netz findet die Aktivierung des Netzes in eine Richtung statt, dabei beginnt dieses bei der Eingabeschicht und führt in Richtung der Ausgabeschicht. Hierbei ähnelt das RNN einem Feed-Forward-Netz, jedoch ersetzt es die einfachen Neuronen eines Feed-Forward-Netzes durch bidirektionale Verbindungen. Diese Verbindung wird rekurrent Neuron oder folgend rekursives Neuron genannt.

Abbildung 19: Einfaches rekurrent Neuron

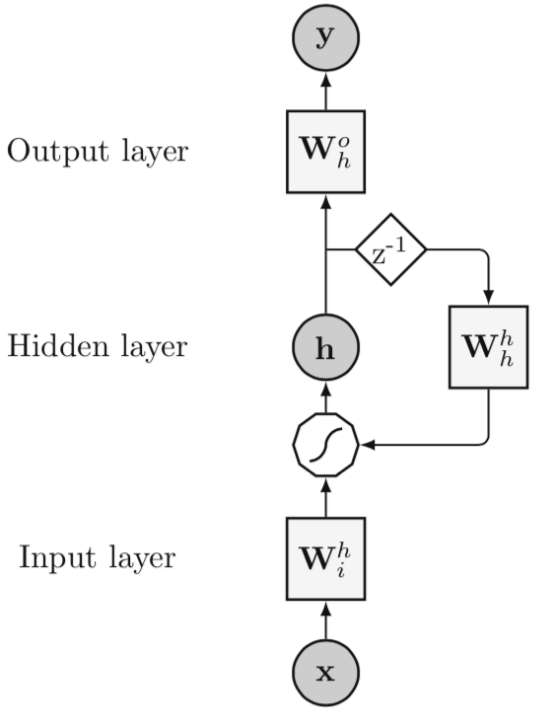


Quelle: [15]

Ein rekursives Neuron ist in der Lage, sich selbst über die Zeit mehrfach auszuführen. Somit erhält das rekursive Neuron Eingaben, produziert Ausgaben und schleust diese an sich selbst weiter. Um ein simples RNN zu erstellen, wird dafür nur ein rekursives Neuron benötigt. Die Abbildung 19 zeigt das rekursives Neuron über eine Zeitachse auf. Dieser Vorgang wird das Aufrollen des Netzes über die Zeitachse genannt. Bei jedem der aufgeführten Ausführungszeitpunkte erhält das rekursive Neuron die Eingabe **x**, sowie die eigene Ausgabe **y**. Dabei können die Eingaben von **x** und **y** gewichtet werden. Zum Zeitpunkt **t** besitzt das Neuron, eine Funktion aller Eingaben der vorhergingen Schritte. Die Abbildung 20 stellt eine einfache RNN Architektur mit drei unter-

schiedlichen Layern dar. Die Quadrate $\mathbf{W_i^h}$, $\mathbf{W_h^h}$, $\mathbf{W_h^o}$ repräsentieren die Matrizen, welche für die Gewichtung der Input-, Hidden-, Outputs Layer zuständig sind [27]. Durch die Eigenschaften des rekurrent Neurons lässt sich ein Gedächtnis imitieren. Der Teil des Netzes, welcher seinen Zustand über mehrere zeitliche Schritte beibehält, wird als Gedächtniszelle bezeichnet [15].

Abbildung 20: Einfache RNN Architektur

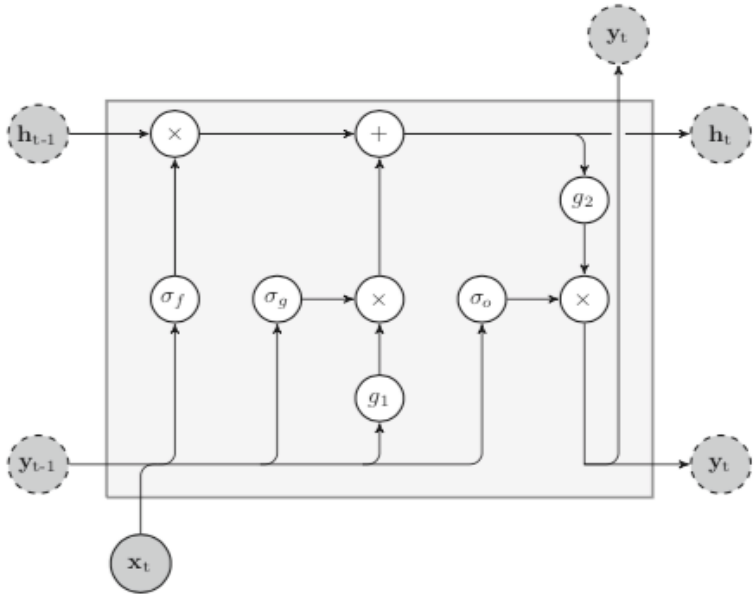


Quelle: [27]

Eine spezielle Form dieser Gedächtniszelle ist dabei die GRU Zelle. Diese besitzt forget und input gates. Diese werden innerhalb der GRU Zelle zu einem Einzigen update gate kombiniert. Das update gate kontrolliert außerdem, wie viele Informationen in den Hidden Layer behalten werden soll. In Abbildung 21 sind die hell grauen Kreise mit durchgezogener Linie die Variablen, welche ihren Inhalt mit dem Eingang und Ausgang des Netzes austauschen. Die dunkel grauen Kreise mit den gestrichelten Linien sind die internen Zustandsvariablen, diese tauschen ihren Inhalt mit den Zellen des Hidden Layers aus. Der Operator *g* ist eine nicht lineare Transformation, die Implementierung findet dabei durch einen hyperbolischen Tangens oder auch Tanh aus Abbildung 10 bekannt statt. Die Kreise mit dem Inhalt **+**, **-1** und **x** stellen lineare Operationen

dar. Die Kreise mit **σr** und **σu** repräsentieren die Sigmoid Funktion, welche in den update gates verwendet werden [27].

Abbildung 21: GRU Zelle



Quelle: [27]

Die meisten erfolgreichen NLP Anwendungen, wie beispielsweise maschinelle Übersetzung oder automatische Zusammenfassung bauen auf RNNs auf. Zunächst aber sollte die Repräsentation von Wörtern geklärt werden. Bei der Darstellung von Wörtern in einem One-Hot-Vektor werden vorhandene Wörter in einem Satz oder Dokuments mit einer **1** gekennzeichnet und fehlende mit **0**. Angenommen es ist ein Vokabular mit 10 Wörtern vorhanden, wie in Tabelle 4. Dabei ist der Eingabesatz *Die Suche nach den richtigen Quellen ist nicht immer einfach.* Dieser Satz würde in dem One-Hot-Vektor folgendermaßen dargestellt werden:

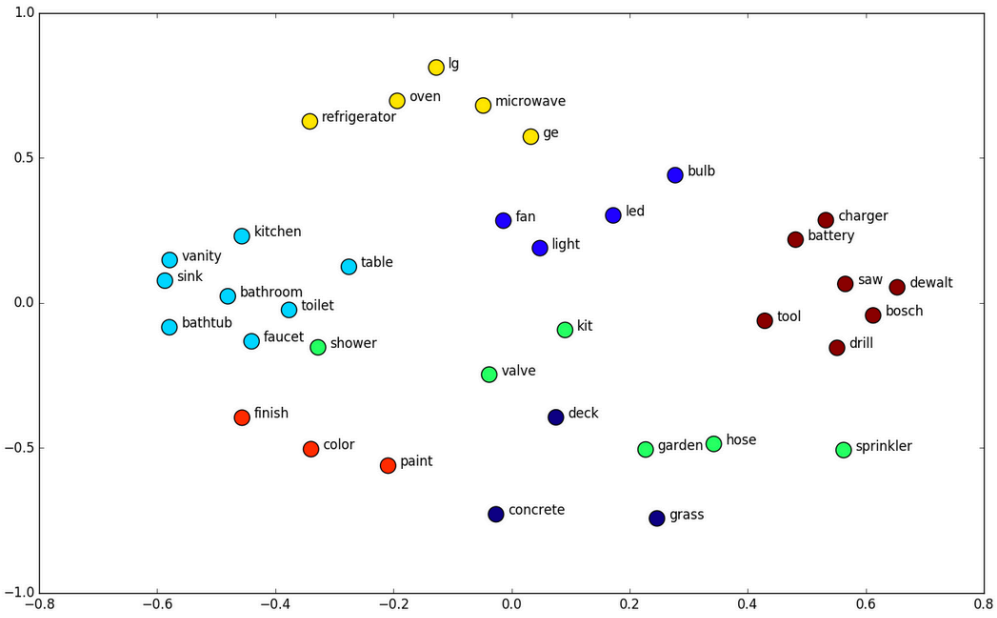
Tabelle 4: One-Hot-Vektor

	Funke	Quellen	Schuhe	immer	nein	ja	einfach	malen	Hallo	Auto
1	0	1	0	1	0	0	1	0	0	0

Quelle: Eigene Darstellung

Dieser Beispielsatz soll eine Demonstration für eine triviale Anwendung des One-Hot-Vektors sein. Bei einem Vokabular mit 1000 Wörtern könnten über 990 Stellen mit einer 0 gefüllt sein. Durch die aufwendige Berechnung vieler Nullstellen ist die Schlussfolgerung, dass diese Repräsentation überaus ineffizient ist. Deswegen gibt es den Ansatz, Wörter die Ähnlichkeiten besitzen, in einem Vokabular dementsprechend ähnlich zu repräsentieren. Wenn als Beispiel *Ich mag die Farbe Grün* als richtig interpretiert wird, soll der Satz *Ich mag die Farbe Rot*, entsprechend ebenfalls als richtig interpretiert werden. Oft wird dafür Word Embedding verwendet. Embedding ist eine Repräsentation eines Wortes mithilfe eines dichten und kleinen Vektors, mit einer Dimension von beispielsweise 150.

Abbildung 22: 2-D Word Embedding



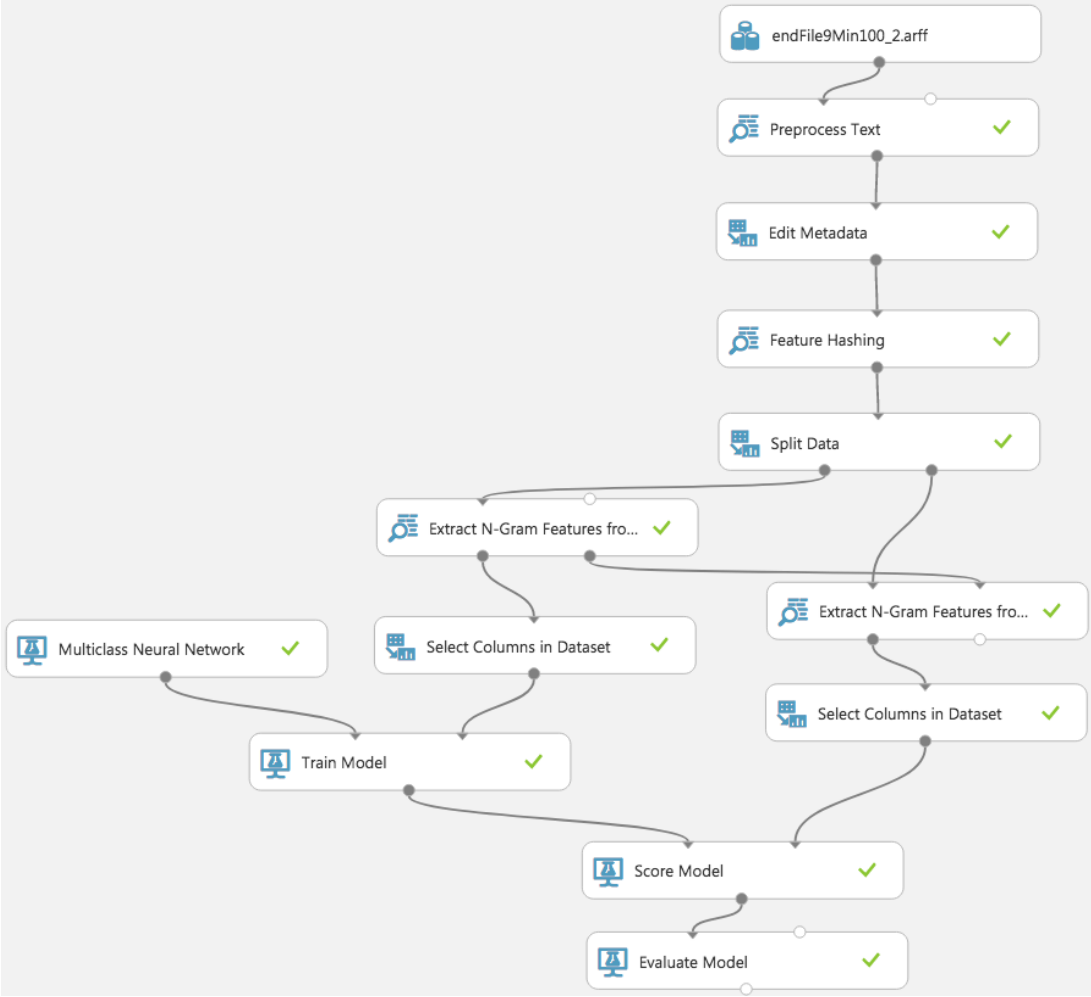
Quelle: [28]

Dieser Vektor wird dem neuronalen Netz zum Trainieren übergeben, um eine bestmögliche Darstellung des Wortes zu erhalten. Beim Beginn des Trainings im neuronalen Netz wird erst einmal willkürlich ein Platz für diesen Vektor ausgesucht. Im Verlauf des Trainings wird dieser durch Backpropagation im Netz weiter verbessert. Nach der Fertigstellung sollten ähnliche Wörter, nah beieinanderliegen, wie zum Beispiel das Word *toilet* und *Bathroom* in der Abbildung 22 aufzeigt [15]. Das weitere Verfahren, um Texte oder Dokumente zu klassifizieren, ist aus den vorherigen Abschnitten bekannt.

2.9 State of the Art

In diesem Unterkapitel sollen drei State of the Art Techniken zum Erstellen einer Baseline herangeführt werden. Zunächst wird mithilfe von Microsoft Azure Machine Learning Studio ein Workflow mit einem neuronalen Netz zur Klassifizierung erstellt. Hierbei handelt es sich um ein einfaches neuronales Netz mit einer Schicht und 57 Neuronen. Es wird der Datensatz 3 der Tabelle 7 für die Klassifizierung verwendet. Im Teil des *Preprocess Text* werden alle notwendigen Schritte zur Vorbereitung des Textes angewandt. Diese werden im Kapitel Datensatz Pre-processing genauer erläutert. Weiterhin wird ein N-Gram Feature mit der Größe von 3 erstellt und TF-IDF verwendet.

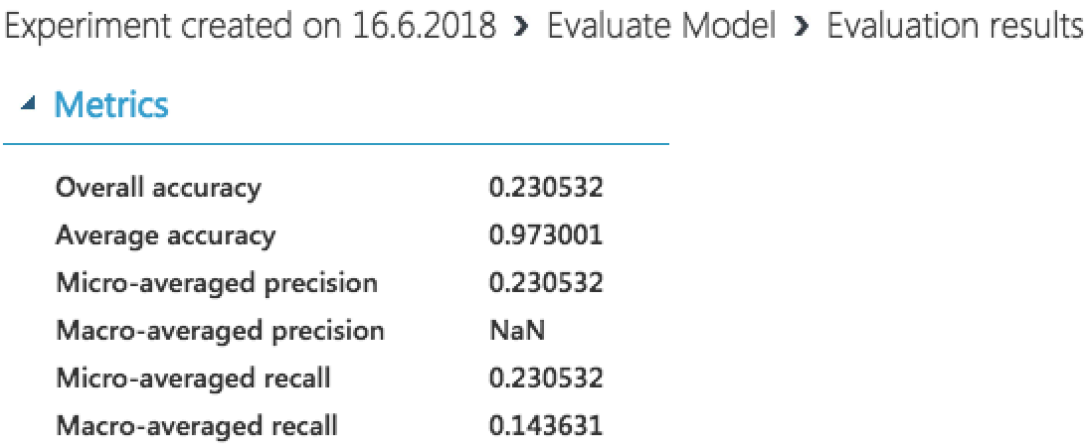
Abbildung 23: Azure Workflow



Quelle: Microsoft Azure Machine Learning Studio Workflow

Das Machine Learning Studio bietet den Vorteil, dass ohne viel Fachwissen, erst Versuche erstellt werden können. Dabei kommt der leichte Umgang der Plattform ersten Test zugute. In Abbildung 24 wird das erzielte Ergebnis vorgestellt. Das einfache neuronale Netz, konnte eine Genauigkeit über alle 57 Sachgruppen von 23,05 % erreichen. Das Ergebnis *NaN* bei Macro-averaged precision kam zustanden, in dem fünf Klassen mit jeweils 0 Instanzen richtig klassifiziert wurden. Somit wurde bei der Berechnung für die Macro-averaged precision mit 0 multipliziert. Trotzdem demonstriert das Ergebnis, dass das Machine Learning Studio gut für erste Tests ist, jedoch nicht an das gewünschte Resultat herankommt.

Abbildung 24: Evaluation Azure Workflow



Quelle: Microsoft Azure Machine Learning Studio Workflow

Eine weitere State of the Art Technologie ist die Naive Bayes Klassifizierung. Diese Klassifizierung wurde mithilfe des Weka Tools erzeugt. Weka ist ein Open Source Tool, welches eine Ansammlung von Machine Learning Algorithmen und Data Mining Ansätze bereitstellt. Ebenso beinhaltet Weka Tools für Data Pre-processing, Klassifizierung, Regression, Clustering et cetera. Das Weka Tool wurde auf der Programmiersprache Java entwickelt. Dabei ist Weka auch gut zur Entwicklung neuer Machine Learning Schemata geeignet [29]. Mit der Naive Bayes Klassifizierung konnte eine Genauigkeit von 45,47 % erreicht und15506 Instanzen korrekt klassifiziert werden.

Tabelle 5: Naive Bayes Klassifizierung

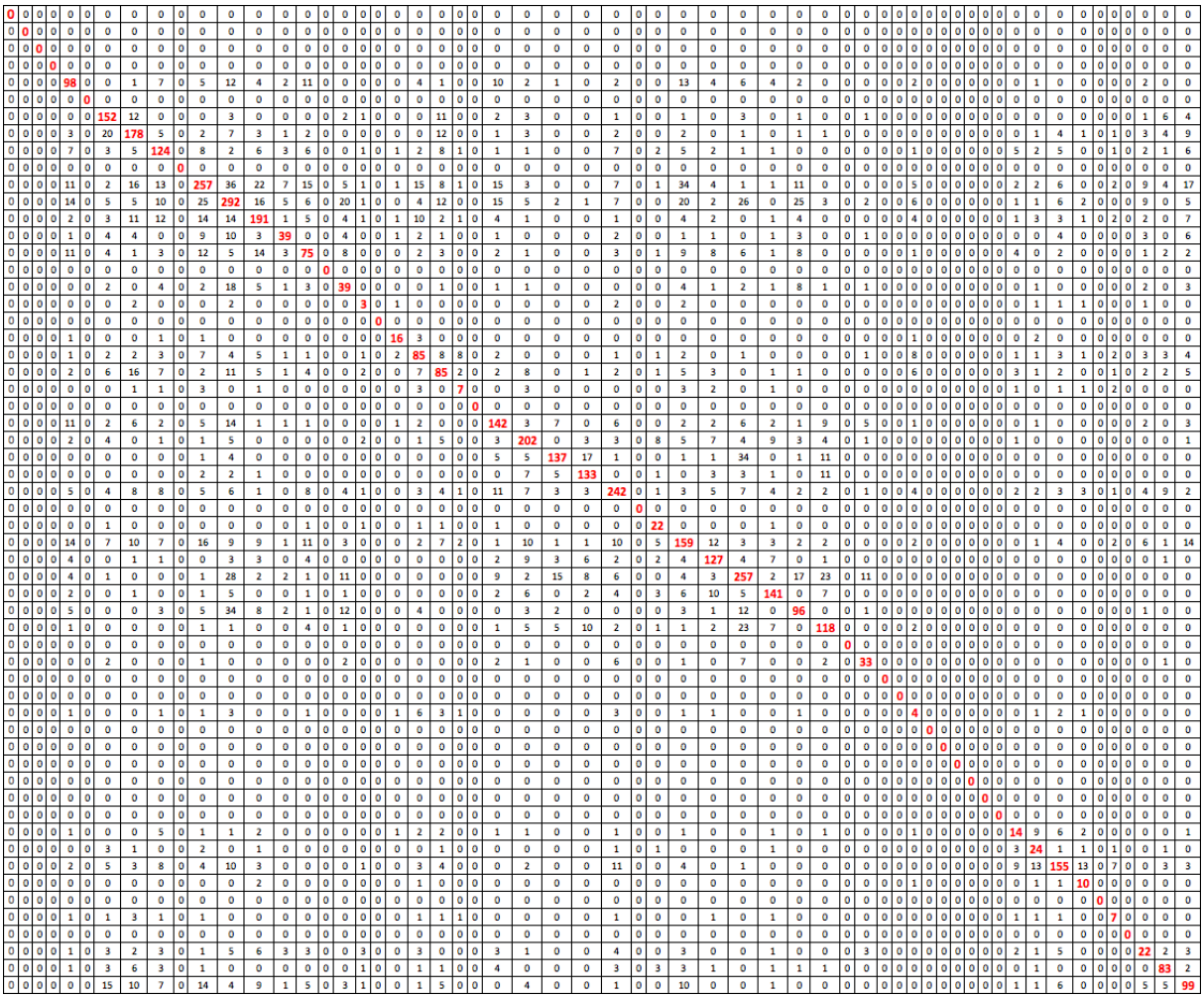
Stratified cross - validation		
Summary		
Correctly Classified Instances	**15506**	**45,4761 %**
Incorrectly Classified Instances	**18591**	**54,5239 %**
Kappa statistic	0,4386	
Mean absolute error	0,0211	
Root mean squared error	0,1158	
Relative absolute error	61,6626 %	
Root relative squared error	88,5076 %	
Total Number of Instances	**34097**	

Quelle: Weka Software

Im Anschluss der Naive Bayes Klassifizierung, wurde ein Test mit einem RNN Netz durchgeführt. Dafür wurde das Hierachical Deep Learning for Text Classification (HDLTex) mit einer Hierarchieebene verwendet. HDLTex wird ausführlich im Kapitel 3.3 erläutert. Das Modell wurde mit 50 Epochen trainiert. Dabei wurden in der Testphase 6820 Instanzen klassifiziert, von denen 3868 Instanzen korrekt klassifiziert worden. Somit wurde eine Accuracy von 56,72 % erreicht. Die Confusion Matrix der RNN Klassifizierung wird in der Abbildung 25 illustriert und zeigt auf, wie gut der Klassifizierer gearbeitet hat. Als Abschluss sollte noch ein Test, mit zwei RNN Netzen gestalten werden, die alle 57 Sachgruppen auf der ersten und zweiten Hierarchieebene klassifizieren. Dieser wird als *HDLTex mit zwei flachen Klassifizierer über die Hierarchieebenen* in der Tabelle 6 mit 50 und 100 Epochen aufgeführt. Dabei werden die gleichen Einstellungen, wie für den Test mit dem RNN Netz verwendet. Die Accuracy mit 50 Epochen lag dabei bei 33,88 % und mit 100 Epochen bei 33,58 %.

Insgesamt wurden drei verschiedene State of the Art Klassifizierer und ein gesonderter Test mit dem HDLTex Tool zur Erstellung einer Baseline verwendet. Anhand dieser Baseline, werden im Kapitel 3 die aufgeführten Tools evaluiert.

Abbildung 25: RNN Klassifizierer



Quelle: Eigene Darstellung

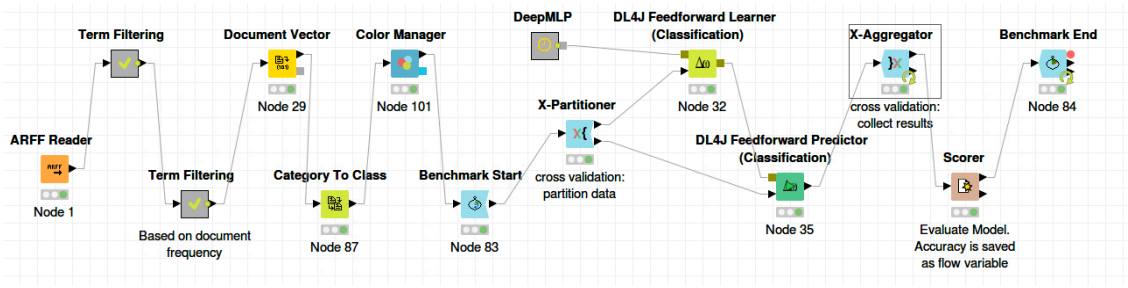
2.10 KNIME Analytics Platform

Die Open Source Software KNIME Analytics Platform ist ideal für Data Science Applikationen und Services gedacht. Durch die dauerhafte Integrierung neuer Entwicklungen im Bereich Data Science, bietet KNIME eine offene Plattform für Standard Benutzer und Entwickler. Durch die intuitive Benutzung dieser Plattform ist eine schnelle Einarbeitung für Anfänger sowie Fortgeschrittenen möglich. Dabei können mit dieser Plattform Data Science Workflows kreiert und zur Wiederverwendung mit der Gemeinschaft geteilt werden. Mit der visuellen Darstellung von Arbeitsschritten durch Icons, die aneinandergeknüpft werden, lassen komplizierte Prozesse ohne Erstellung eines Quellcodes durchführen. Weiterhin bietet die *KNIME Analytics Platform* eine Integrierung mehrere Codierungssprachen wie R, Python et cetera. an. Dabei lassen sich

Verknüpfungen zu externen Datenbanken erstellen oder unterschiedliche Datensätze in Form von JSON, CSV et cetera miteinander und ohne großen Aufwand kombinieren [30].

In der Abbildung 26 wird der Workflow in KNIME dargestellt. Die Vorverarbeitungsschritte des Textes werden in den Icons *Term Filtering* durchgeführt. In diesem Workflow wird eine fünffache Cross Validation für die Trainings- und Testphase verwendet. Das Icon *DeepMLP* beinhaltet drei verdeckte Schichten mit jeweils 100 Neuronen. Es wird ein einfaches DNN verwendet, welches über Backpropagation trainiert wird.

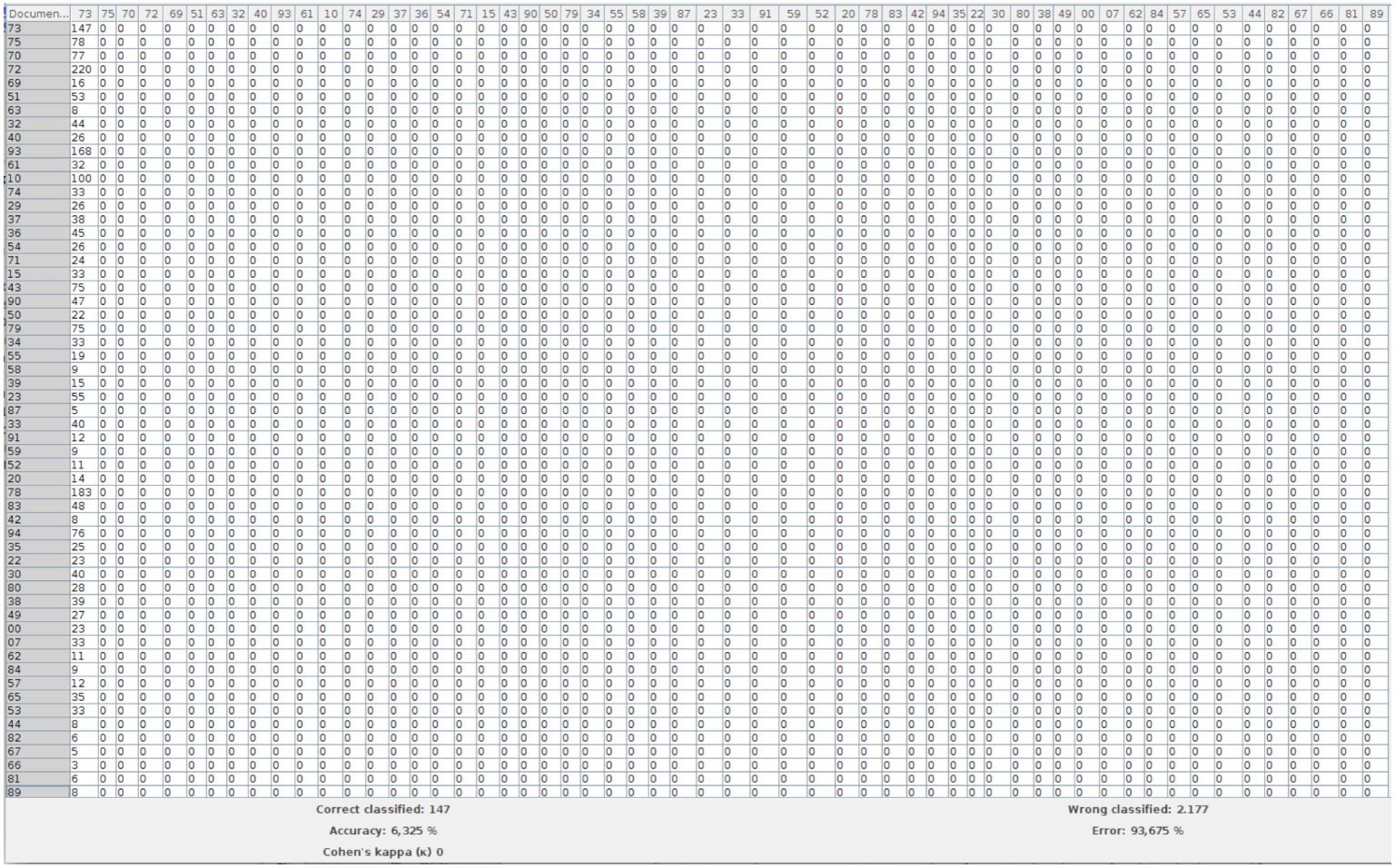
Abbildung 26: KNIME Workflow



Quelle: KNIME Software

In der Abbildung 27 wird dazu die Confusion Matrix mit der Accuarcy von 6,325 % angezeigt. Als mögliche Fehlerursache, könnte einer der zahlreichen Vorverarbeitungsschritte in Betracht gezogen werden, jedoch ist dies nur eine Hypothese. Aus dem zeitlichen Aspekt wurde dies jedoch nicht weiterverfolgt. Die Platform KNIME Analytics wurde über einen längeren Zeitraum benutzt, jedoch konnte kein besseres Ergebnis mit einem DNN erzeugt werden. Wie in der Abbildung 27 zu sehen ist, gibt es keine eindeutige Diagonale in der Confusion Matrix. Die Klassifizierung fand vollständig anhand des Labels 73 statt und alle Instanzen wurden dieser Sachgruppe zugeordnet. Die Fehlerursache ist dabei unbekannt geblieben.

Abbildung 27: KNIME Workflow Confusion Matrix



Documen...	73	75	70	72	69	51	63	32	40	93	61	10	74	29	37	36	54	71	15	43	90	50	79	34	55	58	39	87	23	33	91	59	52	20	78	83	42	94	35	22	30	80	38	49	00	07	62	84	57	65	53	44	82	67	66	81	89
73	147	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
70	77	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
72	220	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
69	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
51	53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
63	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
93	168	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
61	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
74	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
54	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
71	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	75	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
90	47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
79	75	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
55	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
58	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
87	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
91	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
59	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
52	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
78	183	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
83	48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
94	76	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
80	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
49	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
62	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
84	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
57	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
65	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
53	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
82	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
67	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
66	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
81	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
89	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Correct classified: 147

Accuracy: 6,325 %

Cohen's kappa (κ) 0

Wrong classified: 2.177

Error: 93,675 %

Quelle: KNIME Software

Trotz der geringen Accuracy sollte dieses Tool für weitere Arbeit nicht außer Acht gelassen werden. Zum Aufbau eines Grundverständnisses und der benötigten Vorverarbeitungsschritte ist das Tool KNIME gut geeignet. Weiterhin könnten bessere Ergebnisse mit anderen Datensätzen erreicht werden. Für den Einsatz in zukünftigen Arbeiten sollten jedoch alle Einstellungen noch einmal überprüft und gegebenenfalls korrigiert werden.

3 Related Work

Dieses Kapitel soll den aktuellen Stand der Technik im Bereich Verarbeitung textueller Daten und Vorbereitung für Klassifizierung aufzeigen. Für die folgende Auswahl galt dabei das Kriterium, dass die verwendeten Werkzeuge nicht älter, als zwei Jahre alt sein dürfen. Es sollen zwei neue Techniken und die momentan verwendete Lösung der DNB aufgeführt werden. Am Schluss des Kapitels werden die Ergebnisse der aufgeführten Tools evaluiert.

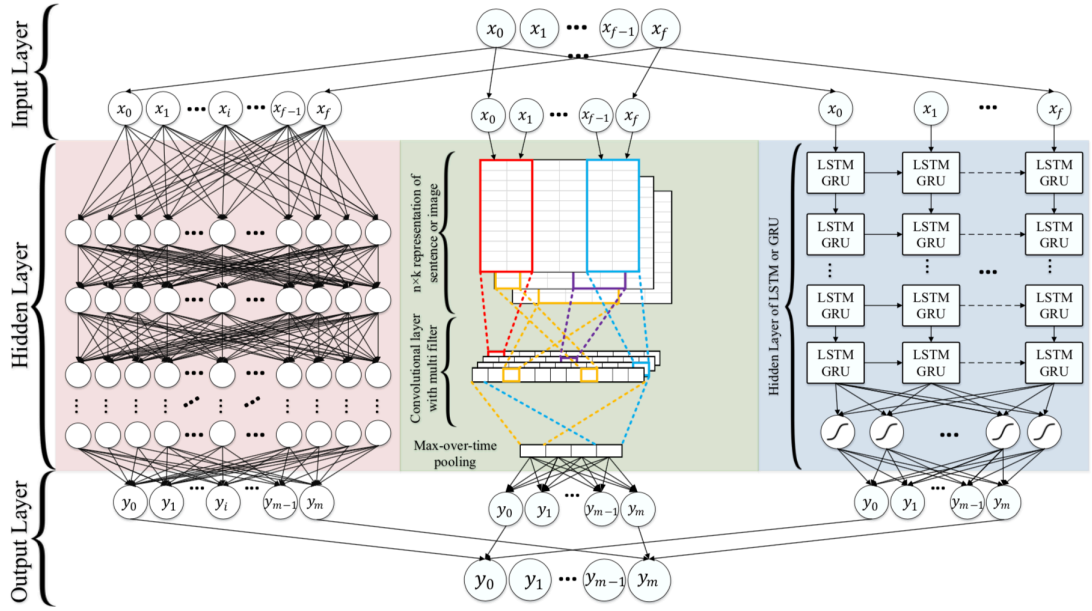
3.1 Averbis Extraction Platform

Zum Einsatz kommt die kommerziell verwendete Software *Averbis Extraction Platform* bei der DNB. Hierzu liegen nicht viele Informationen vor, doch es sollen die Eckdaten beschrieben werden. Für die automatische Erschließung der Netzpublikationen soll ein Wert von mindestens 80 % richtig kategorisierter Publikationen erreicht werden. Dabei muss erwähnt werden, dass zwei Sachgruppen von der Software aufgeführt werden. Hierbei spielt es jedoch keine Rolle, ob beide aufgeführten Sachgruppen richtig klassifiziert wurden. Die erkannten Unterschiede zwischen intellektueller und automatischer Erschließung bedeuten nicht unbedingt, dass die Kategorisierung vollkommen falsch ist. Bei Themenstellungen, die in unterschiedlichen Bereichen handeln, können auch mehrere DDC-Sachgruppen korrekt sein. Dabei wird die Klassifizierung als richtig angesehen, sobald eine von beiden Sachgruppen korrekt klassifiziert ist. Problematisch wird es, da die DNB nur eine DDC Hauptsachgruppe vergibt [8]. Hier liegt ein Konflikt vor und dieser führt sehr wahrscheinlich zu falsch Klassifizierungen. Es sollte hinterfragt werden, wie weit dieser Wert als gut anzusehen ist. Durch die ungleiche Verteilung der Instanzen je Sachgruppe konnte hierbei eine starke Differenz zwischen den Sachgruppen vorliegen. Die geforderten mindestens 80 %, könnten nur bei den stärkeren besetzen Sachgruppen vorliegen. Diese Hypothese ist jedoch nicht nachvollziehbar, weil diesbezüglich keine weiteren Informationen vorliegen. Weiterhin ist bekannt, dass die Software *Averbis Extraction Platform* SVMs zur Klassierung verwendet. Damit lässt sich sagen, dass das Unternehmen Averbis eine State of the Art Lösung verwendet.

3.2 Random Multimodel Deep Learning for Classification

Das folgende Tool ist eine Kombination aus mehreren State of the Art Deep Learning Ansätzen zur Klassifizierung. Random Multimodel Deep Learning for Classification (RMDL) soll dabei den Benutzer unterstützen, die beste Deep learning Struktur und Architektur zu finden. RMDL verbessert die Robustheit und die Genauigkeit, durch Kombination verschiedener Netze. Der Vorteil des RMDL Tools liegt darin, dass es mit unterschiedlichen Inhalten umgehen kann. Es können Videos, Bilder, Symbole oder Texte zur Klassifizierung verwendet werden. Die Vorverarbeitung der Daten wird anhand der eingespielten Daten automatisch durchlaufen. Im RMDL Tool sind die Netze DNN, RNN und CNN integriert. Somit kann benutzerspezifisch die Kombination der Netze und deren Anzahl gesteuert werden. Jedes verwendete Netz des RMDL wird als RDL bezeichnet. Für die Klassifizierung einer Instanz wird das *majority vote* Verfahren benutzt. Hier gibt jedes RDL eine Vorhersage für die Instanz ab und die Mehrheit entscheidet, zu welcher Klasse diese Instanz klassifiziert wird [31]. Durch gute Resultate, die mit dem Datensatz 2 der Tabelle 7 erzielt wurden, ist dieses Tool in die nähere Auswahl gekommen. Die Genauigkeit des RMDL Tools lag bei 72,95 % und das mit drei DNN Netzen. Somit wäre bereits eine Verbesserung zu den State of the Art Lösungen gefunden.

Abbildung 28: RMDL Architektur



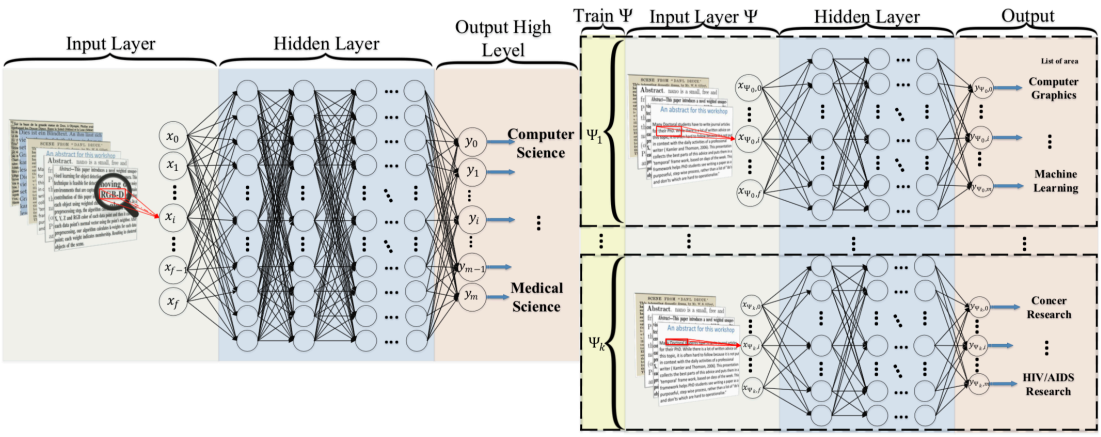
Quelle: [31]

Die Abbildung 28 soll die Verarbeitung der unterschiedlichen Netze darstellen und die schlussendliche Klassifizierung anhand aller vorhandenen Netze.

3.3 Hierachical Deep Learning for Text Classification

Als Nächstes wird das HDLTex Tool beschrieben. HDLTex wurde von den Entwicklern des RMDL Tools erstellt und ist dessen Vorläufer. Der Grundgedanke bestand darin, eine hierarchische Text Klassifizierung zu erstellen. Mit dem HDLTex Tool ist es möglich, eine Klassifizierung über zwei Hierarchieebenen durchzuführen. Durch eine Anpassung des Quellcodes könnten weitere Ebenen hinzugefügt werden. Auch bei diesem Tool sind die State of the Art Netze DNN, CNN und RNN integriert. Dabei lässt sich eine Kombination aus den genannten Netzen gestalten, um das beste Resultat, auf den verwendeten Datensatz zu generieren. Es können neun Kombinationen mit den Netzen durchgeführt werden. Dabei ist es zu empfehlen, auf das RNN Netz zurückzugreifen. Durch die natürliche Beschaffenheit dieses Netzes ist es optimal für Texte geeignet. In der ersten Hierarchieebene wird zunächst die Hauptkategorie klassifiziert und in der zweiten Hierarchieebene die Unterkategorie. Die Hauptkategorie wird in der Tabelle 6 als L1 und die Unterkategorie als L2 aufgeführt. Der Vor- und gleichzeitig der Nachteil dieses Tools ist dabei, die Hierarchie. Der Datensatz sollte möglichst viele Instanzen für das Training besitzen. Nach der Klassifizierung auf die Hauptkategorien werden die Trainingsdaten für die Unterkategorien geteilt und es bleiben wenig Instanzen für die einzelnen Unterkategorien übrig [32]. Mit HDLTex gelang es, die Genauigkeit der Vorhersagen basierend auf dem Datensatz 2 der Tabelle 7 zu verbessern. In der ersten Hierarchieebene L1 wurde eine Genauigkeit von 82,84 % und auf der zweiten Hierarchieebene L2 auf 85,92 erreicht.

Abbildung 29: HDLTex Architektur



Quelle: [32]

In der Abbildung 29 werden die zwei Hierarchieebenen von HDLTex illustriert. Dabei wird aufgeführt, dass zunächst eine Instanz anhand der ersten Hierarchieebene klassifiziert wird. Als Beispiel könnte hier die Hauptkategorie *Computer Science* dienen. Im nächsten Schritt wird dieselbe Instanz, basierend auf der Hauptkategorie *Computer Science*, auf die Unterkategorie klassifiziert. Dabei könnte die Unterkategorie zu *Computer Graphics* oder *Machine Learning* gehören. Jedoch sollte das obere Ergebnis kritisch hinterfragt werden. Die zweite Hierarchieebene L2 darf keinen höheren Wert als L1 aufweisen. L2 hat einen höheren Wert, weil die Klassifizierungen von L1 und L2 unabhängig voneinander gewertet wird. Jedoch sollte die Klassifizierung als richtig angesehen werden, wenn beide Hierarchieebenen korrekt klassifiziert werden. Um diese Abhängigkeit zu realisieren, wurde der Quellcode von HDLTex erweitert. In der Tabelle 6 wird das abhängige Resultat aufgelistet und es wurden die Epochen 50, sowie 100 verwendet.

3.4 Evaluation der Ergebnisse

In diesem Unterkapitel soll kurz aufgeführt werden, was für Ergebnisse Vorliegen und welches Tool für folgende Tests verwendet wird. In der Tabelle 6 werden alle vorhandenen Ergebnisse aufgeführt. Alle Ergebnisse basieren auf dem Datensatz 3 der Tabelle 7. Es werden alle bis dahin erzeugten Ergebnisse aufgelistet.

Tabelle 6: Evaluation der Ergebnisse

Name	Korrekt klassifiziert
Azure Workflow	23,05 %
Naive Bayes Klassifizierung	45,47 %
RNN Netz (50 Epochen)	56,72 %
RMDL (100 Epochen mit zwei RDLs)	48,34 %
RMDL (100 Epochen mit neun RDLs)	57,47 %
HDLTex (50 Epochen)	L1 = 70,89 % L2 = 56,49 %
HDLTex (100 Epochen)	L1 = 70,90 % L2 = 57,41 %
HDLTex mit zwei flachen Klassifizierer über die Hierarchieebenen (50 Epochen)	L1 = 57,71 % L2 = 33,88 %
HDLTex mit zwei flachen Klassifizierer über die Hierarchieebenen (100 Epochen)	L1 = 57,74 % L2 = 33,58 %

Quelle: Eigene Darstellung

An der Tabelle 6 ist ersichtlich, dass das HDLTex Tool zwar nicht das beste Resultat liefert, jedoch nur marginal schlechter als das RMDL Tool performt. Die Entscheidung fiel trotzdem auf das HDLTex Tool, um die hierarchische Struktur des Datensatzes und das DDC System zu nutzen. Die State of the Art Lösung mit der Naive Bayes Klassifizierung sollte jedoch nicht ganz außer Acht gelassen werden. Durch das schnelle Training und die schnelle Klassifizierung dieses Tools könnte viel Rechenzeit, auf Kosten von Accuracy gespart werden. Weiterhin konnte festgestellt werden, dass das einfache RNN Netz fast gleichwertig wie das RMDL Tool performt. Durch die wesentlich schnellere Trainingsphase sollte diese Option nicht vernachlässigt werden. Das RMDL Tool wurde einmal mit zwei RDLs ausgeführt, die aus zwei hintereinander geschalteten RNN Netzen bestand. Weiterhin wurden neun RDLs getestet, die aus einer Kombination aus drei DNN, drei RNN und drei CNN Netzen entstand. Der Test mit dem HDLTex Tool und den zwei flachen Klassifizierer, sollte als weitere Richtlinie für die Ergebnisse dienen. Das HDLTex Tool mit dem hierarchischen Ansatz performt dabei besser, als mit zwei flachen Klassifizierer die alle 57 Sachgruppen berücksichtigen.

4 Erstellung der Datensätze

In diesem Kapitel geht es um die Datensätze, die während dieser Arbeit Verwendung fanden. Es wird zunächst beschrieben, wie der Datensatz eins und zwei erstellt wurden. Im Laufe der Arbeit kamen Verbesserungen der Erstellung des dritten Datensatzes zugute. Dies spiegelte sich in der vereinfachten Nachbereitung des Datensatzes wider.

Tabelle 7: Datensätze

Datensatz	Nr	Instanzen
Reduzierter Datensatz (1000 Instanzen pro Sachgruppe, 80 Sachgruppen insgesamt)	1	79224
Kompletter Datensatz auf Deutsch & mit Filterung (80 Sachgruppen)	2	2274056
Kombination von Features im Datensatz (57 Sachgruppen)	3	34097
WOS5736	4	5736
WOS11967	5	11967
Roher Datensatz mit verschiedenen Sprachen & ohne Filterung	6	6392367

Quelle: Eigene Darstellung

In der Tabelle 7 finden sich alle verwendeten Datensätze wieder. Zur Erstellung der Datensätze wurden alle Titeldaten der DNB bereitgestellt. Die ursprünglichen Daten im XML-Format betrugen 86,76 GB. Insgesamt waren 6392367 Instanzen im Datensatz 6 der Tabelle 7 vorhanden, die in unterschiedlichen Sprachen vorlagen. Durch eine Umformatierung in das ARFF Format und Filterung auf deutsche Texte sowie Texte, die eine DDC Sachgruppierung besitzen, konnte der Datensatz auf 534,5 MB beziehungsweise auf 2274056 Instanzen reduziert werden. Hierfür wurde eine XSLT Datei in Kombination mit einem Python Skript von Herrn Prof. Heß bereitgestellt. In der XSLT – Datei ist die Aneinanderreihung der Tags entscheidend für den entstehenden Datensatz. In folgenden Quellcode wird die XSLT – Datei illustriert.

```
1     <?xml version="1.0"?>
2
3     <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:m="http://www.loc.gov/MARC21/slim">
4
5     <xsl:template match="/">
6     <arff>
7     <xsl:for-each select="//m:record[@type='Bibliographic']"><xsl:if
      test="m:datafield[@tag='245']"><xsl:if test="m:data-
      field[@tag='082']"><xsl:if test="m:datafield[@tag='041']/m:sub-
      field[@code='a']='ger'">'<xsl:for-each select="m:data-
      field[@tag='245']/m:subfield[@code='a']"><xsl:value-of se-
      lect="."/></xsl:for-each><xsl:for-each select="m:data-
      field[@tag='245']/m:subfield[@code='b']"><xsl:value-of se-
      lect="."/></xsl:for-each>', '<xsl:value-of select="m:data-
      field[@tag='082']/m:sub-
      field[@code='a']"/>'</xsl:if></xsl:if><xsl:text>&#xa;</xsl:text></xsl
      :if></xsl:for-each>
8     </arff>
9
10    </xsl:template>
11
12    </xsl:stylesheet>
```

Es dürfen keine Leerzeichen oder Abstände nach dem Beginn des ersten „<xsl:for-each>“ – Tag hinzugefügt werden.

Um die erzeugte Arff – Datei abzuschließen, musste ein Header in die Datei hinzugefügt werden:

```
1     @RELATION Hochschulschriften
2
3     @ATTRIBUTE Titel string
4     @ATTRIBUTE Sachgruppe string
5
6     @DATA
```

Der Header gibt wieder, welche Attribute in der Datei vorhanden sind und um welchen Datentyp es sich handelt. Mit @Data wird der Beginn der Instanzen angegeben. Weiterhin mussten die Sachgruppen auf Richtigkeit überprüft werden. Eine korrekte Sachgruppe könnte theoretisch von 000 bis 999 gehen, wobei nicht alle Sachgruppen momentan vergeben sind. Einige Sachgruppen in der komprimierten Arff Fassung hatten beispielsweiße die Sachgruppe „11b“

oder „10c“. Diese Sachgruppen können nicht zur Klassifizierung verwendet werden. Die gerade erwähnten Sachgruppen wurden mithilfe eines Python Codes entfernt. Somit ist ein Datensatz mit den möglichen Sachgruppen von 000 bis 999 geblieben. Daraufhin musste auch eine Bearbeitung bzw. Filterung der Titeldaten geschehen. Oft erkennbar war, dass es trotz Filterung auf deutsche Titel weiterhin Titel in Italienisch oder Französisch dabei waren. Es wurde eine weitere Filterung der Zeichen è, é, à, ò, ù, ì, á, `, ó durchgeführt, um möglichst viele Fremdsprachen aus dem Datensatz zu entfernen. Nach den Filterprozessen sollten nur noch vereinzelte Titel mit einer Fremdsprache vorhanden sein. Eine Filterung auf englische Wörter wäre äußerst schwierig und durch vorkommende Anglizismen wie beispielsweise „Machine Learning“ oder „Data Mining“ könnte bei Entfernung dieser Wörter der Datensatz verfälscht werden. Es soll am folgenden Ausschnitt aufgezeigt werden, wie die Instanzen in dem Datensatz zustande kommen.

```
1   <arff xmlns:m="http://www.loc.gov/MARC21/slim">
2   'Malakologische Abhandlungen', '590'
3   'Technisch-wissenschaftliche Abhandlungen der Osram-Gesellschaft', '600'
4   'Zoologische Abhandlungen', '590.5'
5   'Annual abstract of statistics / Central Statistical Office', '310'
```

Somit konnte der Datensatz an die 3552000 Instanzen vorweisen. Zum Trainieren ist der Datensatz jedoch ungeeignet groß. Für erste praktische Tests wurden die Sachgruppen auf jeweils eine Stelle reduziert. Somit wurde die Instanz: 'Malakologische Abhandlungen', '590' zu 'Malakologische Abhandlungen', '5' umgewandelt. Weiterhin wurden die Instanzen drastisch reduziert. Für die ersten Trainingsläufe wurden jeweils 200 - 2000 Instanzen aus den Sachgruppen 0 – 9 verwendet. Dieser Datensatz soll für erste Test mit dem Neuronalen Netzwerk dienen. Ziel hierbei ist die Beschaffung erster Ergebnisse und Optimierung der Parameter für das neuronale Netz. Der Datensatz wies weiterhin Verbesserungspotenzial auf und wurde noch mal bereinigt. Der vordere Teil der Datei wurde abgeglichen, somit konnten doppelte Einträge entfernt werden. Dabei wurden 6590 weitere Einträge gelöscht. Titel die nur Zahlen

beinhalteten wie beispielsweise '1969', '830' oder '30 + 50 + 100' sind dabei entfernt worden.

Zur Verbesserung der Klassifizierung wurden weiter Features in den Datensatz eingebaut. Innerhalb des Attributs Titel der Arff Datei wurden die Features Untertitel, Mitwirkende und die Hochschule beziehungsweise die Universität hinzugefügt. Damit kann eine zukünftige Publikation des gleichen Autors einfacher klassifiziert werden. Auch durch Nutzen der Hochschule beziehungsweise der Universität wird dies vereinfachen. Weiterhin sollte mit dem Datensatz 3 eine Mindestqualität implementiert und alle Sachgruppen mit weniger als 100 Instanzen entfernt werden. Um alle genannten Features in den Datensatz 3 zu integrieren, schrumpfte die Anzahl der vorhandenen Instanzen enorm. Der Datensatz 3 hat eine gesamte Größe von 6,7 MB, was im Vergleich zum vorherigen Datensatz 1 mit über 120 MB sehr gering ist. Dies wurde erreicht, in dem alle Sachgruppen ohne mindestens 100 Instanzen je Sachgruppe entfernt wurden. Als Maximum kann eine Sachgruppe 1000 Instanzen besitzen. Alle weiteren Instanzen, die zu dieser Sachgruppe gehören wurden entfernt. Durch die Selektierung der Sachgruppen konnte ein gewisses Maß an Datenqualität hergestellt werden, jedoch ist die Verwendung der Accuracy mit Vorsicht zu genießen. Durch die ungleichmäßige Verteilung der Instanzen kann die Accuracy nicht als Qualitätsmerkmal dienen.

In der Tabelle 8 werden alle Sachgruppen aufgezeigt, die mindestens 100 und maximal 1000 Instanzen aufweisen. Insgesamt sind 57 Sachgruppen im Datensatz 3 zur Klassifizierung übriggeblieben. Dabei beinhalten 18 Sachgruppen lediglich an die 100–200 Instanzen. Lediglich 20 Sachgruppen können eine Anzahl von 1000 Instanzen aufweisen.

Tabelle 8: Sachgruppen mit Instanzenanzahl

Sachgruppe	Instanzen	Fachgebiet
0	1000	Allgemeines, Wissenschaft
7	605	Nachrichtenmedien, Journalismus, Verlagswesen
10	793	Philosophie
15	1000	Psychologie
20	137	Religion, Religionsphilosophie
22	296	Bibel
23	1000	Theologie, Christentum
29	152	Andere Religionen
30	1000	Sozialwissenschaften, Soziologie, Anthropologie
32	1000	Politik
33	1000	Wirtschaft & Natürliche Ressourcen, Energie und Umwelt
34	1000	Rech
35	344	Öffentliche Verwaltung & Militär
36	901	Soziale Probleme, Sozialdienste, Versicherungen
37	1000	Erziehung, Schul - und Bildungswesen
38	550	Handel, Kommunikation, Verkehr
39	117	Bräuche, Etikette, Folklore
40	225	Sprache, Linguistik
42	122	Englisch
43	904	Deutsch & Andere germanische Sprachen
44	136	Französisch, romanische Sprachen allgemein
49	169	Andere Sprachen
50	190	Naturwissenschaften
51	922	Mathematik
52	198	Astronomie, Kartografie
53	1000	Physik
54	1000	Chemie
55	966	Geowissenschaften
57	1000	Biowissenschaften, Biologie
58	302	Pflanzen (Botanik)
59	634	Tiere (Zoologie)

61	1000	Medizin, Gesundheit
62	1000	Ingenieurwissenschaften und Maschinenbau & Elektrotechnik, Elektronik & Ingenieurbau und Umwelttechnik
63	1000	Landwirtschaft, Veterinärmedizin
65	1000	Management
66	1000	Technische Chemie
67	1000	Industrielle und handwerkliche Fertigung
69	277	Hausbau, Bauhandwerk

70	519	Künste, Bildende Kunst allgemein
71	145	Landschaftsgestaltung, Raumplanung
72	890	Architektur
73	417	Plastik, Numismatik, Keramik, Metallkunst
74	102	Grafik, angewandte Kunst & Comics, Cartoons, Karikaturen
75	278	Malerei
78	915	Musik
79	1000	Freizeitgestaltung, Darstellende Kunst & Öffentliche Darbietungen, Film, Rundfunk & Theater, Tanz & Spiel & Sport

80	127	Literatur, Rhetorik, Literaturwissenschaft
81	105	Englische Literatur Amerikas
82	179	Englische Literatur
83	1000	Deutsche Literatur & Literatur in anderen germanischen Sprachen
84	160	Französische Literatur
87	114	Lateinische Literatur
89	148	Literatur in anderen Sprachen & Slawische Literatur

90	163	Geschichte
91	187	Geografie, Reisen & Geografie, Reisen (Deutschland)
93	708	Alte Geschichte, Archäologie
94	1000	Geschichte Europas & Geschichte Deutschlands

Quelle: Eigene Darstellung

5 Datensatz Pre-processing

In diesem Kapitel werden die wichtigsten Verarbeitungskomponenten aufgezählt und erläutert. Es soll ein rudimentäres Verständnis für die einzelnen Komponenten aufgebaut werden. Es besteht hier lediglich eine Aufzählung von den möglichen Verarbeitungskomponenten und keine Rheinfolge. Diese werden je nach Lernalgorithmus benötigt oder auch nicht. Als Beispiel kann hier die Term Frequency aufgeführt werden, die nur beim DNN Netz zum Einsatz kam. Bei Verwendung von KNIME, mussten die Verarbeitungskomponenten manuellen hinzugefügt beziehungsweise in den Workflow integriert werden. Beim RMDL und HDLTex Tool liefen alle notwendigen Verarbeitungskomponenten in einem automatisierten Prozess ab.

5.1 Stop Word Filter

Unter einem Stop Word Filter wird ein Filter verstanden, der Wörter aus einem Dokument, Korpus oder Titel entfernt, welche keinen signifikanten Informationsgehalt zur Suche beitragen. Diese Wörter werden durch ihr häufiges Aufkommen nicht in Textindexierungen beachtet [33]. Im Deutschen Sprachgebraucht gelten die Wörter das, *ist*, *ein*, *text*, *und*, *noch* als Stoppwörter [33]. Diese werden auch unter den Kategorien bestimmter und unbestimmter Artikel, Konjunktion, Präposition oder Negation eingegliedert [34]. Diese Wörter werden für die syntaktischen und grammatikalischen Funktionen einer Sprache verwendet und geben daher keinen inhaltlichen Rückschluss auf das zu suchende Dokument [35]. Stop – Wörter werden auch in anderen Sprachen verwendet, wie im Englischen für die Wörter *to*, *has*, *the*, *be*, *or*.

Der Stoppwörter Filter wird weiterhin dazu eingesetzt, um die oben genannten Wörter aus der Indizierung einer Suchanfrage zu entfernen und Anfragen an einer Suchmaschine nicht unnötig zu erschweren. Suchmaschinen müssen diese Wörter filtern, da ansonsten jede Anfrage des Benutzers, durch das häufige Aufkommen dieser Wörter das Ergebnis verfälscht. Der Benutzer würde ansonsten keine genauen Ergebnisse auf spezifische Suchanfragen bekommen.

5.2 Snowball Stemmer

Mithilfe des Stemming Verfahrens, werden Wörter die Ähnlichkeiten vorweisen zu einem Wort zusammengefasst. Als Beispiel wird das Wort Autoreifen verwendet, was zu dem Wort *Auto* reduziert wird. Falls ein Wort wie *Autofarbe* hinzugefügt wird, wird es ebenfalls von dem Suffix gelöst und zu der Grundform umgewandelt. Hierbei geht es um die Wörtervielfalt oder auch Morphologie eines Wortes zu reduzieren. Bei der Indexierung von Wörtern kann das entscheidend sein, um die Komplexität von Sätzen zu reduzieren und bessere Suchergebnisse zu erzielen.

Die Stemming Verfahren werden dabei in drei Hauptkategorien aufgeteilt. Dabei wird in Regelbasiertes -, Statisches oder ein hybrides Stemming unterschieden. Der Snowball Stemmer gehört zu den Regelbasierten Stemmer. Dabei enthält der Snowball Stemmer Regeln, die das Ende eines Wortes entfernen, die Länge eines Wortes festlegen und besondere Zustände eines Wortes festlegen [36]. Schon ein einfacher Ansatz von Stemming und das Dekomprimieren von Wörtern kann eine signifikante Performanceverbesserung bewirken [37]. Zusätzlich bietet der Snowball Stemmer ein deutsches Vokabular, was ein klarer Vorteil und ein Entscheidungsmerkmal ist.

5.3 Bag of Words

Zu den wichtigsten Teilaufgaben der Klassifizierung gehört die Feature-Extraktion und die Selektierung. Wie im Unterkapitel 2.7 schon angeschnitten, gibt es drei Hauptkriterien für eine gute Feature Auswahl. Zunächst muss die Frage geklärt werden, wie das Textdokument am besten als Feature Vektor repräsentiert wird. Dafür kann das Bad of Words Modell verwendet werden, welches sich als Standard im Bereich des NLP etablierte. Zunächst wird ein Vokabular, mit allen vorkommenden Wörtern im Trainingsdatenset erstellt. Jedes Wort wird dabei durch eine Zahl repräsentiert, die angibt, wie oft das Wort vorkam. In dem erzeugten Vokabular werden die Wörter als Elemente dargestellt, wobei die Reihenfolge dieser keine Rolle spielt. Die Bag of Words Repräsentation soll anhand zweier Dokumente vorgestellt werden [19]:

- D_1: “Each state has ist own laws.“
- D_2: “Every country has its own culture.”

Das Vokabular wird zum Kreieren eines d-Dimensionalen Feature Vektors verwendet. Jedes Dokument wird durch ein eigenen Feature Vektor repräsentiert. Die Größe der Dimensionalität wird dabei durch die Anzahl der unterschiedlichen Wörter bestimmt. In der Abbildung 30 kann noch entschieden werden, ob die absolute Anzahl der Wörter aufgeführt wird oder nur eine binäre Anzeige von 0 oder 1 stattfindet. Dies sollte jedoch abhängig vom Lernalgorithmus entschieden werden [19].

Abbildung 30: BoW Vektorisierung

Table 1: Bag of words representation of two sample documents D_1 and D_2.

	each	state	has	its	own	laws	every	country	culture
$\mathbf{x}_{D1}$	1	1	1	1	1	1	0	0	0
$\mathbf{x}_{D2}$	0	0	1	1	1	0	1	1	1
$\sum$	1	1	2	2	2	1	1	1	1

Quelle: [19]

5.4 Term Frequency

Anstatt einer Wiedergabe von 0 oder 1, kann der Ansatz des Term Frequency (TF) verwendet werden. Hierbei wird gezählt, wie oft ein Term in einem Dokument vorkommt. Dabei wird die TF als ganze Zahl zurückgegeben. Oft wird dabei die rohe TF durch Teilung der Dokumentenlänge normalisiert [38].

5.5 Inverse Document Frequency

Durch häufig vorkommende Wörter wie der, die oder das könnte die Wichtigkeit dieser Wörter missinterpretiert werden. Dafür wurde der Ansatz von *Inverse Document Frequency* (idf) entwickelt. Dabei werden Wörter als wichtig erachtet, die seltener in den Dokumenten vorkommen. Somit werden die seltenen auftretenden Wörter stärker gewichtet, als die Wörter, die oft in Dokumenten vorkommen [38].

5.6 Term Frequency – Inverse Document Frequency

Ein weiterer Ansatz ist die *Term Frequency - Inverse Document Frequency* (TF-IDF). Das TF-IDF kann als gewichtete TF angesehen werden und ist besonders nützlich, wenn aus dem Dokumentenkorpus nicht die Stoppwörter entfernt wurden. Dieser Ansatz verfolgt den Gedanken, dass die Wichtigkeit eines Wortes umgekehrt proportional zu der Häufigkeit ist, mit der es in allen

Dokumenten vorkommt. TF-IDF wird häufig zur Sortierung von Dokumenten nach Relevanz in unterschiedlichen Text Mining Aufgaben verwendet.

5.7 PoS Tagging

Mithilfe von Part of Speech (PoS)-Tagging können Nomen, Verben, Adjektive oder Adverbien erkannt werden. Weiterhin können regelbasierte Ansätze verwendet werden. Wichtig hierbei ist es, auch Leerzeichen oder zusammengesetzte Wörter zu erkennen, wie bei *Internet of Things*. Durch Hilfe von PoS-Tagging können weiterhin die umgebenden Wörter analysiert werden und verraten somit die Struktur des Satzes [39].

5.8 N-Grams

Ein sogenanntes N-Gram kann aus jedem beliebigen Satz erstellt werden. Dabei kann ein Wort als Sequenz einer Reihe von n Wörtern gehandhabt werden. Dies ist der einfachste Fall eines N-Grams und wird Unigram genannt. Dabei werden Wörter, Buchstaben oder Symbole als einzelnes Wort betrachtet. Im Fall eines N-Grams sollte auf die Sprache und auf den Anwendungsfall bezogen entschieden werden, wie groß das “N“ ist. In einigen Studien kam heraus das die Größe des “N“ zwischen 4 < n < 8 die beste Genauigkeit für englische Texte aufweist [40]. Weiterhin konnte erkannt werden, dass die Größe 3 oder 4 am besten als anti-spam Filter für E-Mails geeignet war [41].

Abbildung 31: N-Gram Darstellung

- unigram (1-gram):

a	swimmer	likes	swimming	thus	he	swims

- bigram (2-gram):

a swimmer	swimmer likes	likes swimming	swimming thus	...

- trigram (3-gram):

a swimmer likes	swimmer likes swimming	likes swimming thus	...

Quelle: [19]

In der Abbildung 31 werden drei unterschiedliche N-Gram Modelle aufgezeigt. Es wird illustriert, wie sich die Größe des “N“ auf die Teilung des Satzes auswirkt.

5.9 Global Vectors for Word Representation

Global Vectors for Word Representation (GloVe) [42] [43] ist ein unüberwachter Lernalgorithmus, der zur Darstellung von Word Vektoren dient. Dabei wird GloVe für die Netze CNN und RNN verwendet. Die euklidische Distanz hilft dabei die linguistische oder semantische Ähnlichkeit zwischen zwei Wort Vektoren zu finden. Dabei können oft die nächstgelegenen Wort Vektoren seltene jedoch wichtige Wörter offenbaren. Falls das gesuchte Wort *frog* (Frosch) wäre, wären die nächstgelegenen Wörter:

- Frogs
- Toad
- Litoria
- Leptodactylidea
- Rana
- Lizard
- Eleutherodactylus

Interessant hierbei ist, dass die Froschart *Eleutherodactylus* aufgelistet wird, obwohl es mit dem eigentlichen Wort *frog* nicht direkt in Verbindung gesetzt werden könnte [42] [43].

In der Tabelle 9 wird untersucht, wie sich eigen erstellte GloVe Modelle im Vergleich mit dem vortrainierten GloVe Modell schlagen. Dazu wurden zwei Korpora in einfacher Ausführung und mit einem N-Gram verwendet. Dabei wird das GloVe Modell mit der Dimension 100 und 300 ausprobiert. Nach jedem Test wird die Differenz zum Original in der Spalte Differenz aufgeführt. Die Zellen auf der linken Seite, mit einem grauen Hintergrund, zeigen den verwendeten Korpus an. Die folgenden Tests wurde mit dem HDLTex Tool durchgeführt. Dafür wurden pro Durchlauf 50 Epochen verwendet. Weiterhin wurde die Kombination mit einem RNN Netz in der erst Hierarchieebene und einem RNN Netz in der zweiten Hierarchieebene verwendet. In der Tabelle 9 wird die Accuracy als Ausgabewert aufgeführt.

Tabelle 9: GloVe unter Verwendung unterschiedlicher Korpora

	Versuch	Haupt-sach-gruppe	Sach-gruppe 0	Sach-gruppe 1	Sach-gruppe 2	Sach-gruppe 3	Sach-gruppe 4	Sach-gruppe 5	Sach-gruppe 6	Sach-gruppe 7	Sach-gruppe 8	Sach-gruppe 9
VersuchNr / Korpus	**GloVe Englisch**	**0,5996**	**0,8873**	**0,8281**	**0,6813**	**0,5109**	**0,4537**	**0,7263**	**0,6203**	**0,6066**	**0,4761**	**0,6282**
News Korpus												
1	**GloVe V1 Dim_300**	**0,6123**	**0,8789**	**0,8138**	**0,6438**	**0,5174**	**0,5123**	**0,7021**	**0,6148**	**0,6161**	**0,3995**	**0,6259**
	Differenz	**0,0127**	**-0,0084**	**-0,0143**	**-0,0375**	**0,0065**	**0,0586**	**-0,0242**	**-0,0055**	**0,0095**	**-0,0766**	**-0,0023**
2	**GloVe V2 Dim_300**	**0,6139**	**0,8873**	**0,8166**	**0,6375**	**0,5109**	**0,5309**	**0,6841**	**0,6133**	**0,5832**	**0,3971**	**0,6141**
	Differenz	**0,0143**	**0,0000**	**-0,0115**	**-0,0438**	**0,0000**	**0,0772**	**-0,0422**	**-0,0070**	**-0,0234**	**-0,0790**	**-0,0141**
3	**GloVe V1 DIM_100**	**0,6112**	**0,8648**	**0,8109**	**0,6375**	**0,5167**	**0,4691**	**0,7052**	**0,6047**	**0,5885**	**0,4115**	**0,6353**
	Differenz	**0,0116**	**-0,0225**	**-0,0172**	**-0,0438**	**0,0058**	**0,0154**	**-0,0211**	**-0,0156**	**-0,0181**	**-0,0646**	**0,0071**
4	**GloVe V1 Dim_300**	**0,5474**	**0,8060**	**0,8122**	**0,6667**	**0,5025**	**0,4678**	**0,7274**	**0,6510**	**0,5996**	**0,4816**	**0,5983**
	Differenz	**-0,0522**	**-0,0813**	**-0,0159**	**-0,0146**	**-0,0084**	**0,0141**	**0,0011**	**0,0307**	**-0,0070**	**0,0055**	**-0,0299**
5	**GloVe V2 Dim_300**	**0,5457**	**0,8328**	**0,8016**	**0,622**	**0,4975**	**0,4622**	**0,7305**	**0,6421**	**0,6059**	**0,4054**	**0,594**
	Differenz	**-0,0017**	**0,0268**	**-0,0106**	**-0,0447**	**-0,0050**	**-0,0056**	**0,0031**	**-0,0089**	**0,0063**	**-0,0762**	**-0,0043**

Dewiki Korpus												
6	**GloVe V1 Dim_100**	**0,5476**	**0,8239**	**0,8122**	**0,6598**	**0,4866**	**0,4510**	**0,7336**	**0,6551**	**0,6006**	**0,4079**	**0,6371**
	Differenz	**-0,0520**	**-0,0634**	**-0,0159**	**-0,0215**	**-0,0243**	**-0,0027**	**0,0073**	**0,0348**	**-0,0060**	**-0,0682**	**0,0089**
7	**GloVe V2 Dim_100**	**0,5541**	**0,8388**	**0,8122**	**0,6323**	**0,4844**	**0,4762**	**0,7274**	**0,6534**	**0,6006**	**0,4226**	**0,594**
	Differenz	**-0,0455**	**-0,0485**	**-0,0159**	**-0,0490**	**-0,0265**	**0,0225**	**0,0011**	**0,0331**	**-0,0060**	**-0,0535**	**-0,0342**
8	**GloVe V1 Dim_300**	**0,5701**	**0,8358**	**0,8095**	**0,5636**	**0,5112**	**0,451**	**0,7274**	**0,6397**	**0,5996**	**0,4324**	**0,6285**
	Differenz	**-0,0295**	**-0,0515**	**-0,0186**	**-0,1177**	**0,0003**	**-0,0027**	**0,0011**	**0,0194**	**-0,0070**	**-0,0437**	**0,0003**
Ngram-news Korpus												
9	**GloVe V1 Dim_300**	**0,5968**	**0,8845**	**0,8052**	**0,6156**	**0,4841**	**0,5062**	**0,6849**	**0,6086**	**0,5822**	**0,4139**	**0,6424**
	Differenz	**-0,0028**	**-0,0028**	**-0,0229**	**-0,0657**	**-0,0268**	**0,0525**	**-0,0414**	**-0,0117**	**-0,0244**	**-0,0622**	**0,0142**
Ngram-dewiki Korpus												
10	**GloVe V1 Dim_100**	**0,6053**	**0,8986**	**0,8252**	**0,6813**	**0,5065**	**0,5000**	**0,7170**	**0,6234**	**0,5960**	**0,4498**	**0,6212**
	Differenz	**0,0057**	**0,0113**	**-0,0029**	**0,0000**	**-0,0044**	**0,0463**	**-0,0093**	**0,0031**	**-0,0106**	**-0,0263**	**-0,0070**
11	**GloVe V1 Dim_300**	**0,6092**	**0,8930**	**0,8223**	**0,5938**	**0,5007**	**0,5370**	**0,7154**	**0,6383**	**0,5960**	**0,4139**	**0,6094**
	Differenz	**0,0096**	**0,0057**	**-0,0058**	**-0,0875**	**-0,0102**	**0,0833**	**-0,0109**	**0,0180**	**-0,0106**	**-0,0622**	**-0,0188**

Quelle: Eigene Darstellung

Die Tabelle 9 führt zu einem interessanten Ergebnis. Obwohl die getesteten vier Korpora auf Deutsch waren, schnitt das vortrainierte GloVe Modell, welches mit englischen Wörtern trainiert wurde, am besten ab. Lediglich der Test 10 konnte an das originale Ergebnis rankommen. Dieses Ergebnis ist deshalb so interessant, weil es trotz einer anderen Sprache ein besseres Ergebnis erzielte. Es ist jedoch unklar, welcher Korpus für das vortrainierte Modell verwendet wurde und ob dieses ausschließlich mit englischen Wörtern genutzt wurde. Für die folgenden Tests wurde resultierend aus dem Ergebnis, das originale GloVe Modell verwendet.

6 HDLTex Modellentscheidung

In diesem Kapitel werden empirische Tests aufgezeigt, die mit HDLTex erzeugt wurden. Die Ergebnisse werden am Ende des Kapitels zusammengefasst in einer Tabelle ausgegeben. Es soll die beste Kombination aus den unterschiedlichen Netzen gefunden werden. Es werden unterschiedliche Kombinationen aus den Netzen DNN, RNN und CNN getestet. Als Parameter für HDLTex wurden 100 Epochen, eine Batch Größe von 64 für L1 sowie L2 verwendet, eine maximale Sequenzlänge von 250 Wörtern und das Word Embedding mit der Dimension 100 ausgewählt.

Für das DNN Netz kamen 5 Dense Layer mit 100 Neuronen zum Einsatz. Weiterhin wurde ein Dense Layer als Ausgabeschicht mit der Anzahl der Klassen als Neuronen gewählt.

Beim RNN Netz kamen zwei Layer zur Verwendung. Ein Layer mit 100 GRU Zellen und einem Dense Layer als Ausgabeschicht mit der Anzahl der Klassen als Neuronen.

Das CNN Netz verwendete ein Convolution Layer (128, 5), ein Pooling Layer (5), ein Convolution Layer (128, 5), ein Pooling Layer (5), ein Flatten Layer und als Ausgabeschicht ein Dense Layer mit 128 Neuronen.

Die kommenden Tabellen zeigen mehrere Spalten auf. In der Spalte *val_acc* wird die Accuracy der Testdaten aufgeführt. Die Spalte Name repräsentiert, die Kombination der benutzen Netze. Sachgruppe X gibt die erste Hierarchiestufe an. Weiterhin gibt die Spalte *acc* die Accuracy für die Trainingsdaten wieder. Die Spalten *loss* und *val_loss* repräsentieren die Kostenfunktion für Trainings- und Testdaten. Die aufgeführten Ergebnisse wurden mit dem Datensatz 1 der Tabelle 7 erzeugt.

Tabelle 10: Test CNN-CNN

Name	loss	acc	val_loss	val_acc	Sachgruppe	Haupt Acc
CNN_CNN	0,4858	**0,867**	3,1856	**0,5277**	X	**0,5277**
	0,1542	**0,9523**	4,0215	**0,5297**	0	
	0,1225	**0,9436**	1,4067	**0,7463**	1	
	0,1182	**0,9455**	2,3831	**0,6502**	2	
	0,1784	**0,9403**	5,0935	**0,4368**	3	
	0,3581	**0,8559**	2,6463	**0,6314**	4	
	0,2453	**0,9072**	3,2389	**0,593**	5	
	0,1914	**0,9332**	5,1404	**0,4117**	6	
	0,2581	**0,9069**	3,7775	**0,5397**	7	
	0,3803	**0,8599**	3,4659	**0,4964**	8	
	0,1133	**0,9559**	3,22	**0,5934**	9	

Tabelle 11: Test DNN-CNN

DNN_CNN	2,0249	**0,83**	3,3969	**0,6475**	X	**0,6475**
	0,1567	**0,9526**	3,7294	**0,554**	0	
	0,1033	**0,9465**	1,7274	**0,7397**	1	
	0,1442	**0,9427**	2,2394	**0,6675**	2	
	0,172	**0,9435**	4,2679	**0,4302**	3	
	0,3765	**0,8556**	2,3124	**0,6367**	4	
	0,2439	**0,9078**	2,976	**0,5743**	5	
	0,193	**0,9331**	5,0095	**0,4315**	6	
	0,2543	**0,9086**	3,627	**0,5412**	7	
	0,3763	**0,858**	3,4671	**0,4933**	8	
	0,1059	**0,9577**	2,9313	**0,6257**	9	

Tabelle 12: Test RNN-CNN

RNN_CNN	0,4733	**0,8708**	3,5602	**0,5176**	X	**0,5176**
	1,2532	**0,3485**	1,1333	**0,3954**	0	
	0,1418	**0,9386**	1,3638	**0,7364**	1	
	0,1169	**0,9455**	2,1285	**0,6663**	2	
	0,1774	**0,9435**	5,1897	**0,4393**	3	
	0,3545	**0,8583**	2,1268	**0,6202**	4	
	0,2439	**0,9058**	3,0131	**0,5857**	5	
	0,1975	**0,9314**	5,0938	**0,4192**	6	
	0,2581	**0,9047**	3,9681	**0,5347**	7	
	0,3791	**0,8608**	3,7235	**0,4969**	8	
	0,1223	**0,959**	2,7886	**0,6326**	9	

Tabelle 13: Test CNN-RNN

CNN_RNN	2,2565	**0,8109**	3,101	**0,6361**	X	**0,6361**
	0,045	**0,9862**	1,7168	**0,6627**	0	
	0,0415	**0,9833**	0,7762	**0,8369**	1	
	0,0217	**0,9915**	1,4405	**0,7194**	2	
	0,0512	**0,9863**	2,192	**0,5795**	3	
	0,0916	**0,9719**	1,0244	**0,8022**	4	
	0,1022	**0,9635**	1,2607	**0,7363**	5	
	0,0625	**0,9801**	2,086	**0,5948**	6	
	0,0795	**0,9735**	1,554	**0,6942**	7	
	0,1794	**0,9387**	1,9111	**0,6053**	8	
	0,0272	**0,9905**	1,4552	**0,7272**	9	

Tabelle 14: Test DNN-RNN

DNN_RNN	2,4375	**0,7945**	3,2155	**0,6349**	X	**0,6349**
	0,0442	**0,9858**	1,7215	**0,6644**	0	
	0,0419	**0,9824**	0,774	**0,8386**	1	
	0,0213	**0,9925**	1,415	**0,7256**	2	
	0,522	**0,9854**	2,1991	**0,58**	3	
	0,091	**0,9727**	1,0037	**0,8016**	4	
	0,1021	**0,9631**	1,2581	**0,7368**	5	
	0,0628	**0,9811**	2,0945	**0,598**	6	
	0,0799	**0,9734**	1,5433	**0,6922**	7	
	0,1793	**0,9384**	1,9093	**0,6011**	8	
	0,0272	**0,9909**	1,46	**0,7289**	9	

Tabelle 15: Test RNN-RNN

RNN_RNN	0,2267	**0,9291**	1,4428	**0,6767**	X	**0,6767**
	0,0444	**0,9849**	1,8553	**0,6503**	0	
	0,0373	**0,9841**	0,7565	**0,8188**	1	
	0,017	**0,9931**	1,4338	**0,7318**	2	
	0,0568	**0,984**	2,3173	**0,5668**	3	
	0,0865	**0,9728**	0,9439	**0,7937**	4	
	0,0968	**0,9651**	1,2886	**0,7249**	5	
	0,0576	**0,9792**	2,1529	**0,5964**	6	
	0,0808	**0,9745**	1,5743	**0,6902**	7	
	0,1783	**0,9376**	1,8713	**0,6032**	8	
	0,0266	**0,9917**	1,4564	**0,7238**	9	

Tabelle 16: Test DNN-DNN

DNN_DNN	0,1292	**0,9802**	4,5456	**0,4788**	X	**0,4788**
	0,0068	**0,9986**	4,7739	**0,4909**	0	
	X	**1**	2,1347	**0,6562**	1	
	X	**1**	2,8627	**0,6265**	2	
	X	**1**	4,4732	**0,4975**	3	
	0,0063	**0,9959**	2,6511	**0,703**	4	
	0,0091	**0,9963**	3,8141	**0,5567**	5	
	0,0597	**0,9888**	5,3501	**0,4064**	6	
	X	**1**	4,5608	**0,488**	7	
	0,0126	**0,995**	5,2268	**0,4683**	8	
	X	**1**	4,0274	**0,5194**	9	

Tabelle 17: Test RNN-DNN

RNN_DNN	0,4819	**0,869**	3,4654	**0,5184**	X	**0,5184**
	0,0242	**0,9978**	4,5307	**0,6068**	0	
	0,0171	**0,9937**	2,2754	**0,7835**	1	
	0,0067	**0,9988**	2,7959	**0,7443**	2	
	0,0803	**0,9908**	4,3655	**0,5706**	3	
	0,1139	**0,9799**	1,8512	**0,7874**	4	
	0,0871	**0,9839**	2,5663	**0,6776**	5	
	0,037	**0,9947**	4,1817	**0,5453**	6	
	0,0256	**0,9968**	3,768	**0,6459**	7	
	0,0511	**0,9898**	3,5696	**0,5863**	8	
	0,0072	**0,9987**	3,5056	**0,6804**	9	

Der Test in Tabelle 18 konnte aufgrund von Performanceproblemen nicht bis zum Ende durchgeführt werden. Dieser Durchlauf brachte die verwendete Maschine, welche mit 64 GB RAM ausgestattet ist, an den Anschlag und der Prozess wurde vom System abgebrochen. Für die Berechnung wurden über 90 % des RAMs benötigt. Es wurde jedoch das Ergebnis aus der ersten Hierarchiestufe ausgegeben. Dieses Ergebnis kann in den Vergleich miteinbezogen werden.

Tabelle 18: Test CNN-DNN

CNN_DNN	0.4627	**0,8688**	3,3326	**0,53397**	X	**0,53397**

Quelle: Eigene Darstellung

In der Tabelle 19 werden alle Ergebnisse aus der ersten Hierarchiestufe aufgelistet. Dabei ist die Liste nach den besten Ergebnissen, aus der ersten Hierarchiestufe sortiert.

Tabelle 19: Rangliste Ergebnisse

Rangliste:	
RNN_RNN	**0,6767**
DNN_CNN	**0,6475**
CNN_RNN	**0,6361**
DNN_RNN	**0,6349**
CNN_DNN	**0,53397**
CNN_CNN	**0,5277**
RNN_DNN	**0,5184**
RNN_CNN	**0,5176**
DNN_DNN	**0,4788**

Quelle: Eigene Darstellung

Dieses Ergebnis ist als wichtig zu deklarieren, da es die Accuracy für die Hauptsachgruppe der Klassifizierung angibt. Aus diesem Ergebnis kommt hervor, dass das Netz mit der Kombination RNN in der ersten Hierarchiestufe und RNN in der zweiten Hierarchiestufe am besten geeignet ist. In dem kommenden Kapitel wird mit dieser Kombination weitergearbeitet.

7 Kombinierung der Hierarchieebenen

Durch die vorherigen Ergebnisse aus Kapitel 3.4, Kapitel 5.9 und Kapitel 6 ist das zu verwendende Tool als auch die Kombination der Netze bekannt. In diesem Teil der Arbeit soll untersucht werden, ob es möglich ist, eine Verbesserung durch Kombinierung der Vorhersagen aus den beiden Hierarchieebenen zu schaffen. Dabei werden zunächst die Vorhersagen aus der Hierarchieebene 1 genommen und mit den dazugehörigen Vorhersagen aus der Hierarchieebene 2 multiplizieren. Hierbei werden die drei besten Kombinationen, aus den beiden Hierarchieebenen aufgeführt und dem originalen Ergebnis entgegengestellt. Weiterhin wird dieser Ansatz mit der Addition der beiden Hierarchieebenen ausprobiert, um zu sehen, wie weit sich das Ergebnis verbessern oder verschlechtern könnte. Diese Tests werden mit dem Datensatz 3, 4 und 5 der Tabelle 7 durchgeführt.

7.1 HDLTex Quellcode und Einstellungen

In diesem Unterkapitel wird unvollständig der angepassten HDLTex Quellcode erläutert. Dabei muss erwähnt werden, dass es sich hierbei um die Programmiersprache Python handelt.

Mit Zeile 16 lässt sich die Gesamtlänge des einzulesenden Textes angeben. Je nach maximaler Titellänge kann die Zeile 16 variabel eingestellt werden. In der Zeile 17 wird festgelegt, wie viele Wörter behalten werden. Zeile 18 gibt die Dimension des Word Feature Vektors wieder. Dieser muss an die GloVe Version angepasst sein. In Zeile 20 und 21 wird die Batch-Size für die erste und zweite Hierarchieebene angegeben. Die Batch-Size gibt an, wie viele Verarbeitungen parallel laufen. Die Epochenanzahl lässt sich über Zeile 22 einstellen. Um die Netze für HDLTex einzustellen, werden die Zeilen 24 und 25 verwendet. Wie in den vorherigen Kapiteln erwähnt, wird eine Kombination aus den Netzen RNN und RNN verwendet. Dies bedeutet, dass in der ersten Hierarchieebene ein RNN Netz für die Hauptkategorie verwendet wird und weiterhin in der zweiten Hierarchieebene ein RNN Klassifizierung der Subkategorie zum Einsatz kommt.

```
16    MAX_SEQUENCE_LENGTH = 500 # Maximum sequance lenght 500 words
17    MAX_NB_WORDS = 55000 # Maximum number of unique words
18    EMBEDDING_DIM = 100 #embedding dimension you can change it to
19    #{25, 100, 150, and 300} but need to change glove version
20    batch_size_L1 = 64 # batch size in Level 1
21    batch_size_L2 = 64 # batch size in Level 2
22    epochs = 30
23
24    L1_model =2 # 0 is DNN, 1 is CNN, and 2 is RNN for Level 1
25    L2_model =2 # 0 is DNN, 1 is CNN, and 2 is RNN for Level 2
```

Im oberen Quellcode werden alle Einstellungen beschriebenen. In dem folgenden Quellcode wird aufgeführt, wie das RNN Netz erstellt wird.

```
37    def buildModel_RNN(word_index, embeddings_index, nClasses, MAX_SE-
QUENCE_LENGTH, EMBEDDING_DIM):
38          model = Sequential()
39          embedding_matrix = np.random.random((len(word_index) + 1, EM-
BEDDING_DIM))
40          for word, i in word_index.items():
41                embedding_vector = embeddings_index.get(word)
42                if embedding_vector is not None:
43                      # words not found in embedding index will be all-
zeros.
44                      embedding_matrix[i] = embedding_vector
45          model.add(Embedding(len(word_index) + 1,
46                              EMBEDDING_DIM,
47                              weights=[embedding_matrix],
48                              input_length=MAX_SEQUENCE_LENGTH,
49                              trainable=True))
50          model.add(GRU(100,dropout=0.2, recurrent_dropout=0.2))
51          model.add(Dense(nClasses, activation='softmax'))
52          model.compile(loss='sparse_categorical_crossentropy',
53                      optimizer='rmsprop',
54                      metrics=['acc'])
55          return model
```

In der Zeile 50 wurden 100 GRU Zellen eingestellt. Die Zeile 51 ist interessant, da diese die Softmax Aktivierungsfunktion besitzt. Dabei ist zu berücksichtigen, dass der Softmax Layer keine gesonderte Wahrscheinlichkeit für jede Klasse zurückgibt, sondern den maximalen Wert von 1 beziehungsweise 100 % auf die einzelnen Klassen aufteilt. Um das zu erläutern, sollen die Klassen A und B als Beispiel genommen werden. Wenn die Klasse A eine Accuracy von 80 % aufweist, kann die Klasse B nur eine Accuracy von 20 % annehmen.

Dabei ist es irrelevant, wie viele Klassen vorhanden sind. Das Schema passt sich der Anzahl der Klassen an. Weiterhin werden in der Zeile 52 die Kostenfunktion, die Optimierungsfunktion und die Metrik angegeben. Die Metrik gibt die Accuracy der Vorhersage wider.

```
262    ### tmp2 gibt länge von den Vektoren im L2 wieder
263    tmp2 = len(b)
264    for ii in range(len(a)):
265           for tmp3 in range(tmp2):
266                  ### listVek erste Stelle = Multiplikation L1 & L2
267                  ### zweite Stelle welche Stelle in L1
268                  ### dritte Stelle welche Stelle in L2
269                  listVek.append(([float_format-
ter(float(a[ii])*float(b[tmp3][0])),ii,b[tmp3][1]]))
```

Der oben aufgeführte Quellcode illustriert in der Zeile 269, wie die Verrechnung der beiden Hierarchieebenen zustande kommt. Dabei wird in der Zeile 269 in dem Array *listVek* als Beispiel die Werte 0.45, 1, 2 gespeichert. Der erste Wert ist die Vorhersage der Genauigkeit, der zweite Wert gibt die Klasse der ersten Hierarchie zurück und der dritte Wert, gibt die Klasse der zweiten Hierarchieebene wider. Die originale Vorhersage wird anhand zwei Wahrheitsmatrizen dem Benutzer wiedergegeben. Dies soll am Datensatz 4 der Tabelle 7 demonstriert werden. Die Tabelle 20 stellt die Ausgabe der ersten Hierarchieebene wieder.

Tabelle 20: Confusion Matrix erste Hierarchieebene des Datensatz 4

251	2	0
2	**314**	3
2	11	**563**

Quelle: Eigene Darstellung

Hierbei ist gut die Diagonale in der Matrix zu erkennen, was auf eine gute Klassifizierung hinweist. In der Tabelle 21 wird die Klassifizierung anhand der zweiten Hierarchieebene illustriert. Dabei muss die Klassifizierung sowohl in der ersten als auch in der zweiten Hierarchieebene stimmen.

Tabelle 21: Confusion Matrix zweite Hierarchieebene des Datensatz 4

78	1	1	0	0	0	0	0	0	0	0	0
2	**82**	2	0	0	0	0	2	0	0	0	0
2	6	**77**	0	0	0	0	0	0	0	0	0
0	0	0	**0**	0	0	0	0	0	0	0	0
0	0	0	0	**86**	2	8	11	0	1	0	0
0	0	0	0	1	**64**	3	1	0	0	0	0
0	0	0	0	5	3	**53**	4	0	1	0	0
2	0	0	0	3	0	3	**67**	1	0	0	0
0	0	0	0	0	0	1	2	**119**	5	6	1
0	1	0	0	0	0	3	2	8	**115**	9	0
1	0	0	0	0	0	1	2	8	7	**138**	0
0	0	0	0	0	0	0	0	2	0	1	**144**

Quelle: Eigene Darstellung

Auch in dieser Matrix ist die Diagonale gut zu erkennen und es gibt wenig falsche Klassifizierungen. Nach der Ausgabe der originalen Vorhersage werden weiterhin die nächsten drei bis fünf Wahrscheinlichkeiten ausgegeben wie im kommenden Quellcode aufgeführt.

```
370   ### Ausgabe Wahrscheinlichkeiten
371   print(f' Beste Wahrscheinlichkeit')
372   print(f'Correctly classified on top level:   {ncL1}')
373   if L2_model < 5:
374         print(f'Correctly classified on both levels: {ncL2}')
375   print("Confusion matrix for level 1:")
376   print(cm3)
377
378   print('')
379   print(f' 2. beste Wahrscheinlichkeit')
380   print(f'Correctly classified on top level:   {ncL3}')
381   if L2_model < 5:
382         print(f'Correctly classified on both levels: {ncL4}')
383
384   print('')
385   print(f' 3. beste Wahrscheinlichkeit')
386   print(f'Correctly classified on top level:   {ncL5}')
387   if L2_model < 5:
388         print(f'Correctly classified on both levels: {ncL6}')
```

7.2 Multiplizierung der Vorhersagen aus den Hierarchieebenen

Für den Ansatz der Multiplikation wurden die Datensätze 3, 4 und 5 der Tabelle 7 verwendet. Die Tabelle 22 soll für die kommenden Tests als Hilfsmittel dienen. Die jeweiligen Farben Blau, Orange und Gelb geben die erst, zweit und dritt beste Kombination aus den Vorhersagen wieder. Mit der Farbe Grau wird die Differenz zwischen der originalen Vorhersage und der Kombination der besten Vorhersage in Prozent aufgeführt. Die Farbe Grün gibt das originale Ergebnis des Modells wider. L1 steht für die erste Hierarchieebene und L2 für die zweite Hierarchieebene. L1 wird hochgezählt, sobald die Klassifizierung auf der ersten Hierarchieebene stimmt. L2 wird jedoch erst hochgezählt, wenn beide Hierarchieebenen übereinstimmen. Für folgenden zwei Tests sind dabei die Epochen 1, 5, 10, 20, 30 verwendet worden.

Tabelle 22: Legende

Legende			
	1. beste Wahrscheinlichkeit		Differenz Orgi / 1. beste Wahr
	2. beste Wahrscheinlichkeit		Originales Ergebnis
	3. beste Wahrscheinlichkeit		

Quelle: Eigene Darstellung

Tabelle 23: WOS5736 Datensatz Test

Datensatz:	WOS5736	Epoche: 1	Multiplikation L1 & L2	Softmax Layer	1148 instances tested
L1	1023	1023	1016	1002	0,00 %
L2	421	337	310	275	-19,95 %

Datensatz:	WOS5736	Epochen: 5			
L1	1000	1000	1103	1097	0,00 %
L2	673	419	417	421	-37,74 %

Datensatz:	WOS5736	Epochen: 10			
L1	1126	1126	1126	1122	0,00 %
L2	865	544	485	363	-37,11 %

Datensatz:	WOS5736	Epochen:20			
L1	1125	1125	1126	1123	0,00 %
L2	1004	786	392	357	-21,71 %

Datensatz:	WOS5736	Epochen: 30			
L1	1128	1128	1128	1128	0,00 %
L2	1023	821	396	343	-20,47 %

Quelle: Eigene Darstellung

Tabelle 24: WOS11967 Datensatz Test

Datensatz:	WOS11967	Epoche: 1	Multiplikation L1 & L2	Softmax Layer	2394 instances tested
L1	1292	1292	1264	1214	0,00 %
L2	476	266	286	279	-44,12 %

Datensatz:	WOS11967	Epochen: 5			
L1	2168	2168	2158	2162	0,00 %
L2	1171	707	608	565	-39,62 %

Datensatz:	WOS11967	Epochen: 10			
L1	2236	2236	2233	2237	0,00 %
L2	1657	1021	657	553	-38,38 %

Datensatz:	WOS11967	Epochen: 20			
L1	2237	2237	2239	2238	0,00 %
L2	1955	1319	715	501	-32,53 %

Datensatz:	WOS11967	Epochen: 30			
L1	2246	2246	2246	2243	0,00 %
L2	1993	1438	682	501	-27,85 %

Quelle: Eigene Darstellung

In der Tabelle 23 wird der Test mit dem Datensatz 4 aufgezeigt. Für diesen Test standen 1148 Instanzen zur Klassifizierung zur Verfügung. Für die Tabelle 24 kam der Datensatz 5 zum Einsatz, dabei sind 2394 Instanzen zum Test klassifiziert worden. Beide Datensätze konnten keine besseren Ergebnisse vorweisen, als die originale Klassifizierung. Deshalb sollte der Datensatz 3 intensiver getestet werden, vor allem mit den Epochen 1–10. Dabei wurde

der Test mit den Epochen 1–10, 20, 30, 100 ausgeführt. Es kamen dabei 6820 Instanzen zur Klassifizierung zum Einsatz.

Tabelle 25: DNB Datensatz Multiplikationstest

Datensatz:	DNB Datensatz	Epoche: 1	Multiplikation L1 & L2	Softmax Layer	6820 instances tested
L1	2636	2636	2626	2628	0,00 %
L2	782	233	358	391	-70,20 %

Datensatz:	DNB Datensatz	Epochen: 2			
L1	3130	3130	3137	3116	0,00 %
L2	1171	238	465	463	-79,68 %

Datensatz:	DNB Datensatz	Epochen: 3			
L1	3604	3604	3607	3586	0,00 %
L2	1635	381	476	577	-76,70 %

Datensatz:	DNB Datensatz	Epochen: 4			
L1	3773	3773	3779	3724	0,00 %
L2	2019	490	586	583	-75,73 %

Datensatz:	DNB Datensatz	Epochen: 5			
L1	3887	3887	3852	3838	0,00 %
L2	2039	436	629	663	-78,62 %

Datensatz:	DNB Datensatz	Epochen: 6			
L1	4062	4062	4021	4035	0,00 %
L2	2213	480	584	749	-78,31 %

Datensatz:	DNB Datensatz	Epochen: 7			
L1	4184	4184	4156	4139	0,00 %
L2	2455	574	683	728	-76,62 %

Datensatz:	DNB Datensatz	Epochen: 8			
L1	4330	4330	4308	4279	0,00 %
L2	2596	627	773	833	-75,85 %

Datensatz:	DNB Datensatz	Epochen: 9			
L1	4448	4448	4410	4432	0,00 %
L2	2807	841	937	844	-70,04 %

Datensatz:	DNB Datensatz	Epochen: 10			
L1	4495	4495	4454	4479	0,00 %
L2	2919	869	924	836	-70,23 %

Datensatz:	DNB Datensatz	Epochen: 20			
L1	4804	4804	4800	4802	0,00 %
L2	3558	1079	1226	1084	-69,67 %

Datensatz:	DNB Datensatz	Epochen: 30			
L1	4916	4916	2924	4911	0,00 %
L2	3811	1261	1263	1125	-66,91 %

Datensatz:	DNB Datensatz	Epochen: 100			
L1	4799	4799	4799	4799	0,00 %
L2	3876	1529	1328	1076	-60,55 %

Quelle: Eigene Darstellung

Auch mit dem Datensatz 3 konnte keine Verbesserung erzielt werden. Alle Ergebnisse der ersten Vorhersage waren deutlich schlechter als das originale Ergebnis.

7.3 Addition der Vorhersagen aus den Hierarchieebenen

Für den Ansatz der Addition wurde der Datensatz 3 der Tabelle 7 verwendet. Für die Tests sind die Epochen 1, 5, 10, 20, 30 verwendet worden. Dieser Ansatz wurde ergänzend zu der Multiplikation der Hierarchieebenen gewählt.

Tabelle 26: DNB Datensatz Additionstest

Datensatz:	DNB Daten-satz	Epoche: 1	Addition L1 & L2	Softmax Layer	6820 instances tested
L1	2788	2788	2577	2481	0,00 %
L2	871	241	364	347	-72,33 %

Datensatz:	DNB Daten-satz	Epochen: 5			
L1	3949	3949	3872	3810	0,00 %
L2	2192	536	576	729	-75,55 %

Datensatz:	DNB Daten-satz	Epochen: 10			
L1	4503	4503	4458	4426	0,00 %
L2	2914	875	906	831	-69,97 %
Datensatz:	DNB Daten-satz	Epochen: 20			
L1	4811	4811	2812	4811	0,00 %
L2	3523	1106	1249	922	-68,61 %

Datensatz:	DNB Daten-satz	Epochen: 30			
L1	4896	4896	4894	4905	0,00 %
L2	3845	1427	1308	996	-62,89 %

Quelle: Eigene Darstellung

Mit den Epochen 5, 10, 20, 30 konnte jedoch ein leicht besseres Ergebnis erzielt werden, als mit der Multiplikation. Trotzdem sind resultierenden Ergebnisse wesentlich schlechter, als die der originalen Ergebnisse. Aus diesem Grund wurde dieser Ansatz nicht mehr weiterverfolgt.

7.4 Résumé der Tests

Der Ansatz eine Verbesserung durch Kombinierung der besten Vorhersagen für die beiden Hierarchieebenen durchzuführen, ist nicht erfolgreich gewesen. Eine interessante Beobachtung konnte trotzdem dadurch gemacht werden. Die erste Hierarchie wurde bei der erstbesten Vorhersage nicht schlechter als die originale Vorhersage klassifiziert. Die zweite Hierarchieebene jedoch, ist sehr viel schlechter, als die originale Vorhersage. Es konnten jedoch ein weiterer Ansatz daraus gezogen werden. In zukünftigen Tests könnte erprobt werden, ob eine Änderung der Aktivierungsfunktion positiven Einfluss auf das Ergebnis hätte. Durch Veränderung des Softmax Layers, der die höchste Wahrscheinlichkeit von 1 auf alle Klassen aufteilt, könnte ein besseres Ergebnis erzielt werden. Dabei könnte die Sigmoid Aktivierungsfunktion dafür verwendet werden. Diese Aktivierungsfunktion, gibt für jede Klasse eine unabhängige Wahrscheinlichkeit wieder. Weiterhin sollte dabei die *binary crossentropy* als Kostenfunktion genutzt werden.

8 Limitation und Résumé

Im abschließenden Kapitel sollen die erforschten Ergebnisse und die daraus resultierenden Erkenntnisse Revue passieren lassen werden. Ziel ist es, für den Leser Hilfestellungen zu bieten, damit dieser nicht auf ähnliche Problematiken stößt und weiterhin aus fremden Fehlern lernen kann. Zudem soll der Leser auf den Erkenntnissen aufbauen können und daraus eigene Arbeiten entwickeln können.

8.1 Résumé

In dieser Arbeit wurde untersucht, ob es durch neue Ansätze in der Klassifizierung, eine Verbesserung des aktuellen Stands der DNB hervorbringen lässt. Dafür wurden aktuelle Veröffentlichungen und unterschiedliche Tools untersucht. Durch die Verwendung von HDLTex konnte die gewünschte Verbesserung erreicht werden. Weiterhin sollte dazu untersucht werden, ob eine aufbauende Verbesserung realisiert werden konnte. Dazu wurden empirische Tests durchgeführt, die die Frage klärten, ob aufbauend auf die vorhandene Verbesserung, eine Kombination der Vorhersagen der Hierarchieebenen eine weitere Verbesserung hervorbringt. Dies konnte nicht mit dem beschriebenen Ansatz erreicht werden. Trotz des negativen Ergebnisses lassen sich daraus neue Ansätze für kommende Tests beziehungsweise Arbeiten ziehen. Durch Änderung der Aktivierungsfunktion im RNN Netz könnten unabhängige Wahrscheinlichkeiten genutzt werden. Diese könnten eine Veränderung der Ergebnisse hervorrufen. Somit kann gesagt werden, dass trotz eines negativen Ergebnisses, Möglichkeiten für weitere als auch neue Forschungsfragen eröffnet wurden.

8.2 Stärken und Limitation der Arbeit

Durch Limitationen der Hardware wäre eine Nutzung von vorkonfigurierten Cloudmaschinen von Vorteil gewesen. Weiterhin könnte dadurch, die Zeit der Konfigurierung eingespart werden. Speziell im Machine Learning Bereich, sind

die Cloudmaschinen mit den notwendigen Treibern und Applikationen vorkonfiguriert. Selbst bei Neuerstellung einer Maschine wäre diese sofort zum Einsatz bereit.

Bei der Erstellung des Datensatzes hätte am Anfang der Arbeit mehr Zeit investiert werden sollen. Auch die sofortige Nutzung der erforderlichen Attribute hätte zeitliche Ersparnisse zur Folge. Durch verschieden Variationen des Datensets konnten zwar erste Ergebnisse eingeholt werden, jedoch auf Kosten von Zeit und mehrfacher Erstellung der Datensätze.

Die Ergebnisse aus dem HDLTex Paper [32] mit dem Datensatz 4 und 5 konnten nicht rekonstruiert werden. Weiterhin ist zweifelhaft, ob eine hierarchische Klassifizierung ein besseres Ergebnis hervorbringt, als ein flacher State of the Art Klassifizierer. Dies soll in zukünftigen Arbeiten untersucht werden. Ein weiteres Manko sind die aufgeführten Ergebnisse. Bei genauer Betrachtung der HDLTex Ergebnisse wird klar, dass die Accuracy so nicht stimmten kann. Die zweite Hierarchieebene L2 darf kein höheres Ergebnis, als die erste Hierarchieebene aufweisen. Hier wäre es wichtig, die Accuracy der gesamten Vorhersagekette wiederzugeben, die höchstens den Wert von L1 Accuracy * L2 Accuracy aufweisen darf. Deshalb sollte das Ergebnis des Papers angezweifelt werden.

Durch die Kontaktaufnahme zu Herrn Kowsari wurden Tests mit dem Datensatz 2 durchgeführt. Diese Tests konnten wegen Hardwarelimitationen nicht auf den vorhandenen Maschinen durchgeführt werden. Somit ließen sich diese Ergebnisse in die Wahl des Tools mit einfließen. Dies führte auch zu der Entscheidung, das HDLTex Tool zu verwenden.

Besonders positiv war der enge Kontakt zu Herrn Prof. Heß. Durch schnelle Absprachen und Reaktionszeiten konnten Problematiken schnell identifiziert, sowie ausgebessert werden. Besonders am Ende der Arbeit brachte das einen großen zeitlichen Vorteil.

8.3 Ausblick auf zukünftige Arbeiten

In diesem Unterkapitel sollen weitere Thematiken vorgestellt werden, die aufbauend und ergänzend auf diese Arbeiten wirken. Es könnten folgende Themen infrage kommen:

- Semantische Analyse von Kurztexten basierend auf Twitter Daten
- Integrierung und Nutzung von Multi-View Daten, zur Verbesserung der Klassifizierung
- Classifier Chains für Multi-Label Klassifizierung anhand von Kurztexten
- Evaluierung & Implementierung eines DMS, zur Extrahierung, als auch Vorbereitung wichtiger Informationen
- Einsatz von Ensemble Learning Methoden zur Klassifizierung des DNB Datensatzes

Die erste Thematik könnte mithilfe eines neuen Datensatzes, Rückschluss auf die verwendeten Tools dieser Arbeit geben und ob die untersuchten Ansätze, weiterhin interessant zu erproben wären. Weiterhin sind die Texte kompakter gestaltet, was als große Herausforderung zu verstehen ist. Aus den wesentlich kürzeren Texten eines Tweets müssten die vorhandenen Informationen effizienter genutzt werden. Hierbei könnten neue Ansätze zur Nutzung von Features untersucht werden.

Die Nutzung von Multi-View Daten, könnte zu signifikante Verbesserungen bei der Klassifizierung führen. Die DNB bietet weiterhin Normdaten und Personendaten, die als ergänzende Features verwendet werden könnten. Hierzu müsste untersucht werden, wie weit diese Daten miteinander in Verbindung gebracht und wie die benötigten Features extrahiert werden könnten. Durch Nutzung mehrerer Datensätze könnten die Features weitere wichtige Informationen zur Klassifizierung enthalten.

Bei dem Ansatz des Classifier Chains, lässt sich ein Multilabel in ein Singlelabel Problem umwandeln. Dabei können auch State of the Art Lösungen wie beispielsweise Naive Bayes Klassifizierung als binären Klassifizierer benutzen werden. Hierbei wäre interessant, wie sich der Classifier Chain auf kurzen Texten schlägt, besonders mit dem erstellten Datensatz 3. Dieser Ansatz soll den Ergebnissen dieser Arbeit gegenübergestellt werden.

Die Implementierung eines DMS wäre aus dem Aspekt der Daten Extrahierung interessant. Es könnte ein System aufgebaut werden, welches die unterschiedlichen Datensätze verwaltet und eine Verbindung zwischen ihnen erstellt. Weiterhin sollte untersucht werden, wie das DMS diese Informationen bereitstellen kann. Zunächst müssten jedoch die Funktionen aktueller DMS Systeme näher erläutert werden und dabei beschrieben werden, wie weit diese ausbaubar wären.

Der letzte Ansatz wäre das Benutzen von Ensemble Learning, um unterschiedliche Lernalgorithmen für die Klassifizierung zu nutzen. Dabei soll der Datensatz 3 verwendet und dem jetzigen Ergebnis gegenübergestellt werden. Dazu können State of the Art Lösungen als auch neue Ansätze miteinander kombiniert werden. Ein mögliches Beispiel wäre hier, die Verwendung von HDLTex und RMDL mit weiteren State of the Art Klassifizierer. Hier sollte viel Freiraum für neue Ideen gelassen werden, um unterschiedliche Ansätze zu testen.

Durch Erarbeiten dieser fünf Thematiken, könnten vielversprechende Neuerungen im Bereich der Klassifizierung zustande kommen. Zusätzlich könnten die Auswirkungen auf eine hierarchische Klassifizierung untersucht werden. Weiterhin lassen sich die Ansätze dieser Arbeit auf weitere Datensätze überprüfen.

Literaturverzeichnis

[1] L. Oenning, „WirtschaftsWoche," [Online]. Available: https://www.wiwo.de/technologie/papierloses-buero-das-digitale-buero-hat-viele-schwachstellen/13068284-2.html. [Zugriff am 3 7 2018].

[2] N. Walger, „Sammlung von Netzpublikationen erreicht nächste Stufe," *Dialog mit Bibliotheken 2017,* 2 2017.

[3] U. Junger und U. Schwens, „Die inhaltliche Erschließung des schriftlichen kulturellen Erbes auf dem Weg in die Zukunft," *Dialog mit Bibliotheken 2017,* 2 2017.

[4] C. Diebel, „Netzpublikationen – Sammlung, Archivierung und Bereitstellung in der Deutschen Nationalbibliothek," *Dialog mit Bibliotheken 2015,* 1 2015.

[5] S. Uhlmann, „Automatische Beschlagwortung von deutschsprachigen Netzpublikationen mit dem Vokabular der Gemeinsamen Normdatei (GND)," *Dialog mit Bibliotheken 2013,* 2 2013.

[6] E. Mödden und K. Tomanek, „Maschinelle Sachgruppenvergabe für Netzpublikationen," *Dialog mit Bibliotheken 2012,* 1 2012.

[7] „Katalog der Deutschen Nationalbibliothek," [Online]. Available: https://portal.dnb.de/opac.htm;jsessionid=fDCtHqaJpkijqqGzqFCdpuZFfeK9ijw5S9ZBte2o.prod-fly0?method=showFullRecord&currentResultId=%22eduard%22+and+%22franke%22%26any&currentPosition=0. [Zugriff am 11 7 2018].

[8] C. Schöning-Walter, „Automatische Erschließungsverfahren für Netzpublikationen," *Dialog mit Bibliotheken,* 1 2011.

[9] „Dewey-Dezimalklassifikation," [Online]. Available: http://www.ddc-deutsch.de/Subsites/ddcdeutsch/DE/Home/home_node.html. [Zugriff am 11 7 2018].

[10] U. Reiner, „Automatische DDC-Klassifizierung," *Dialog mit Bibliotheken 2010,* 1 2010.

[11] „Dewey Decimal Classification® (DDC®) summaries," [Online]. Available: https://www.oclc.org/en/dewey/features/summaries.html. [Zugriff am 29 Juni 2018].

[12] G. G. Chowdhury, „Natural language Processing," in *Annual Review of Information Science and Technology*, Bd. 37, 2015, pp. 51-89.

[13] J. Schild, „Gedanken zum Problem Homonymie-Polysemie in synchronischer Sicht," in *STUF - Language Typology and Universals*, Bd. 22, Berling, Akademie Verlag GmbH, 1969, p. 354.

[14] E. D. Liddy, „Enhanced Text Retrieval Using Natural Language Processing," *Bulletin of the American Society for Information Science and Technology,* pp. 14-16, 31 1 2005.

[15] A. Géron, Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow, O'Reilly, 2018.

[16] B. Aunkofer, „Data Science Blog," [Online]. Available: https://data-science-blog.com/blog/2017/12/03/ensemble-learning/. [Zugriff am 02 08 2018].

[17] H. Lohninger, „Grundlagen der Statistik," [Online]. Available: http://www.statistics4u.com/fundstat_germ/ee_classifier_performance_metrics.html. [Zugriff am 03 08 2018].

[18] P. Jordan, Wahrscheinlichkeits- und Matrizenrechnung für Sozialwissenschaftler, Bd. 11, Rainer Hampp Verlag, 2016, p. 43.

[19] S. Raschka, „Naive Bayes and Text Classification I - Introduction and Theory," 2014.

[20] I. Rish, „An empirical study of the naive bayes classifier," in *workshop on empirical methods in artificial intelligence*, 2001.

[21] L. M. Rudner und T. Liang, „Automated Essay Scoring Using Bayes' Theorem," *The Journal of Technology, Learning and Assessment,* 2002.

[22] A. Kroll, Computational Intelligence. Eine Einführung in Probleme, Methoden und technische Anwendungen, Berling: Oldenbourg Wissenschaftsverlag, 2014, pp. 221-278.

[23] J. F. Walde, Design Künstlicher Neuronaler Netze, D. Universitätsverlag, Hrsg., Wiesbaden: Gabler Verlag , 2005, p. 2.

[24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat und G. Irving, „TensorFlow: A System for Large-Scale Machine Learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, Savannah, GA, USA, 2016.

[25] „Faltungsmatrix," [Online]. Available: https://docs.gimp.org/de/plug-in-convmatrix.html. [Zugriff am 01 08 2018].

[26] J. Kim, „http://www.joshuakim.io/," Independent Publisher, 2 12 2017. [Online]. Available: http://www.joshuakim.io/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-embeddings/. [Zugriff am 3 7 2018].

[27] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi und R. Jenssen, Recurrent Neural Networks for Short-Term Load Forecasting : An Overview and Comparative Analysis, Berlin: Springer Verlag, 2017, pp. 26-28.

[28] S. Lynn, „Word Similarities / Synonyms,“ [Online]. Available: https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/. [Zugriff am 4 7 2018].

[29] „Weka 3: Data Mining Software in Java,“ [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/. [Zugriff am 02 08 2018].

[30] „KNIME Analytics Platform,“ [Online]. Available: https://www.knime.com/knime-software/knime-analytics-platform. [Zugriff am 02 08 2018].

[31] M. H. D. E. B. K. J. M. a. L. E. B. Kamran Kowsari, „RMDL: Random Multimodel Deep Learning for Classification,“ in *2018 2nd International Conference on Information System and Data Mining ICISDM*, Lake- land, FL, USA, 2018.

[32] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber und L. E. Barnes, „HDLTex: Hierarchical Deep Learn- ing for Text Classification,“ in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2017.

[33] A. Klahold, Empfehlungssysteme: Recommender Systems - Grundlagen, Konzepte und Lösungen, Springer Verlag, 2009, p. 25.

[34] W. Gaus, „Dokumentations- und Ordnungslehre : Theorie und Praxis des Information Retrieval,“ 5. Auflage Hrsg., Springer Verlag, 2005, p. 254.

[35] o.V., „pc-erfahrung.de,“ 22 5 2018. [Online]. Available: https://www.pc-erfahrung.de/nebenrubriken/sonstiges/webdesignwebentwicklung/stoppwortliste.html. [Zugriff am 4 7 2018].

[36] J. Singh und V. Gupta, „Text Stemming: Approaches, Applications, and Challenges,“ *ACM Computing Surveys ,* Bd. 49, Nr. 45, p. 46, September 2016.

[37] J. Goldsmith, „Unsupervised learning of the morphology of a natural language,“ *Computational Linguistics,* Nr. 27, p. 155, Juni 2001.

[38] C. D. Manning und H. Schütze, Foundations of Statistical Natural Language Processing, Cambridge: MIT Press, 1999.

[39] D. Jurafsky und J. H. Martin, Speech and Language Processing, 3 Hrsg., 2017, pp. 142-167.

[40] V. Keselj, F. Peng, N. Cercone und C. Thomas, „N-gram- based author profiles for authorship attribution.,“ in *In Proceedings of the conference pacific association for computational linguistics*, PACLING, 2003.

[41] I. Kanaris, K. Kanaris, I. Houvardas und E. Stamatatos, „Words versus character n-grams for anti-spam filtering.,“ *International Journal on Artificial Intelligence Tools,* Bd. 16, Nr. 06, p. 1047–1067, 2007.

[42] J. Pennington, R. Socher und C. D. Manning, „GloVe: Global Vectors for Word Representation,“ [Online]. Available: https://nlp.stanford.edu/projects/glove/. [Zugriff am 4 7 2018].

[43] J. Pennington, R. Socher und C. D. Manning, „GloVe: Global Vectors for Word Representation,“ Stanford, 2014.

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

_____

[Ort, Datum Name]

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Master-Thesis unterstützt und motiviert haben.

Zuerst gebührt mein Dank Herrn Prof. Dr. Andreas Heß und Herrn Prof. Dr. Holger Ziekow, die meine Master-Thesis betreut und begutachtet haben. Für die hilfreichen Anregungen und die konstruktive Kritik bei der Erstellung dieser Arbeit möchte ich mich herzlich bedanken.

Ebenfalls möchte ich mich bei Dirk Hölscher bedanken, der mir mit viel Interesse, Ideen und Hilfsbereitschaft zur Seite stand.

Meinen Freunden Andrius Oliveira, Eugen Jesipow, Familie Prochota, Familie Gampp und ganz besonders Annabella Gampp danke ich für den starken emotionalen Rückhalt über die Dauer meines gesamten Studiums.

Abschließend möchte ich mich bei meiner Familie bedanken, die mir mein Studium durch ihre Unterstützung erst ermöglichten und stets ein offenes Ohr für meine Sorgen hatten.

Alexander Gerling

Furtwangen, 30.08.2018

A. [Anhang]

Alle wichtigen Informationen, Quellcodes sowie Literatur befinden sich digital auf der beiliegenden CD.